TOWARDS DISENTANGLEMENT AND COLLABORATION IN OPEN-ENDED DEEP RESEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-ended deep research agents must balance two competing demands: exploring diverse information sources and exploiting retrieved evidence to synthesize coherent reports. Existing systems entangle both capabilities within a single agent, leading to self-evaluation bias, premature convergence, and exploration collapse – the agent settles on a mediocre outline because it cannot objectively assess its own gaps. We introduce **AgentDisCo**, a **Dis**entangled and **Co**llaborative architecture that separates information exploration and exploitation into two specialized agents: a *critic agent* that dialectically probes the evolving outline and proposes gap-aware search queries, and a *generator agent* that retrieves evidence and refines the outline. The disentangled design enables the critic to construct blueprints with adaptive per-key-point retrieval budgets, preventing context-window overload and ensuring balanced evidence coverage. We formalize this alternating interaction as a dual-agent cooperative MDP and show that it admits a *bi-level potential game* structure: in the inner game, the critic and generator alternately refine the outline for each query under a shared quality objective; in the outer game, their respective harnesses are alternately optimized – each holding the other fixed – toward the same end-to-end outline quality, encouraging steady improvement at both levels. Experiments on DeepResearchBench, DeepConsult, and DeepResearchGym show that AgentDisCo consistently outperforms leading deep research agents. We further introduce **GALA** (**G**eneral **A**I **L**ife **A**ssistants), a benchmark for authentic daily-life research needs, and a rendering agent for visually rich report presentations.

1 INTRODUCTION

Open-ended deep research agents capable of synthesizing web-scale information into comprehensive, well-cited reports have emerged as a critical frontier for large language models (LLMs). Closed-source commercial offerings such as GPT Deep Research (OpenAI, 2025a) and Gemini Deep Research (Research, 2025) provide neither technical reports nor open implementations, while a growing body of open technical reports (Han et al., 2025; Li et al., 2025; Lei et al., 2025) has begun to reveal architectural design choices. Yet these systems share a common architectural limitation: they entangle *information exploration* – planning and issuing search queries – with *information exploitation* – synthesizing retrieved evidence into structured outlines – inside a single monolithic agent. This entanglement triggers two well-documented failure modes. First, when the same model both generates and evaluates its own outline, *self-evaluation bias* (Panickssery et al., 2024; Park & Choi, 2026) causes the agent to overrate its current output and under-explore alternative directions. Second, concentrating exploration and exploitation in one policy encourages *exploration collapse*: the agent converges on a narrow set of search strategies, neglecting information sources that fall outside its initial query distribution.

In this paper, we propose **AgentDisCo**, a **Dis**entangled and **Co**llaborative agentic architecture that addresses these failure modes by separating exploration and exploitation into two specialized agents. A *critic agent* dialectically probes the evolving outline, identifies coverage gaps, and proposes targeted search queries; a *generator agent* retrieves evidence for those queries and refines the outline accordingly. This separation yields three concrete benefits: (i) The critic can objectively assess gaps without the self-evaluation bias inherent in judging one's own generation; each agent can be independently optimized for its specific role, enabling clearer credit assignment; (ii) The dialectical interplay between the two agents actively counteracts exploration collapse by continually surfacing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

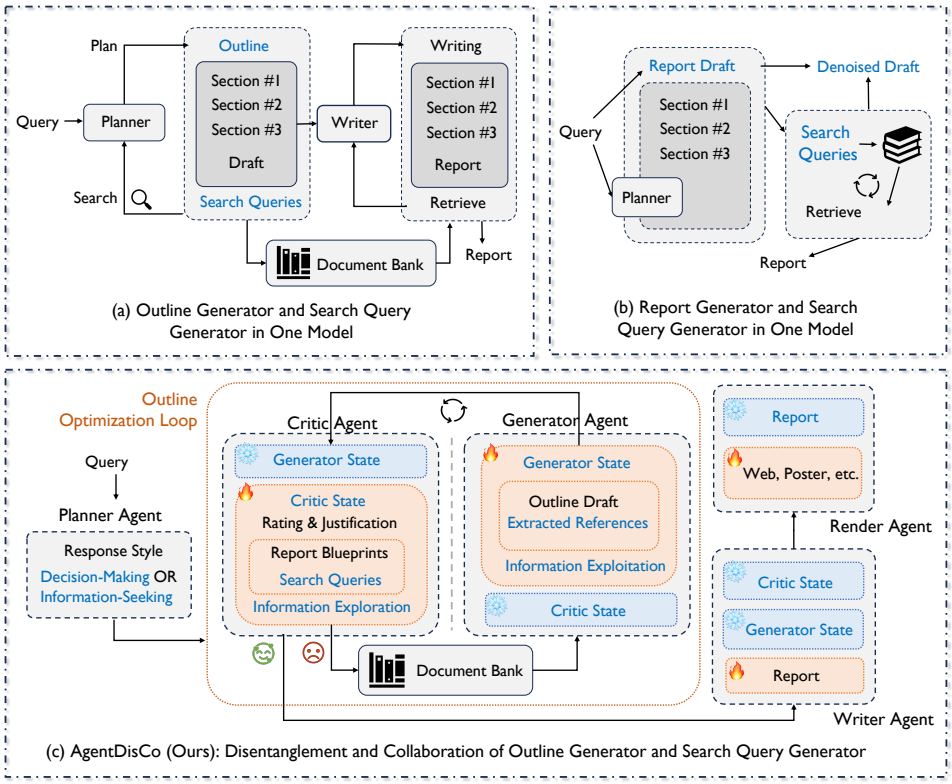

Figure 1: **Comparison of deep research paradigms.** (a) The outline-iterative-optimization paradigm couples outline generation and search query formulation within a single model; (b) the report-iterative-optimization paradigm similarly entangles report generation with search query formulation; (c) in contrast, **AgentDisCo** disentangles the outline generator and the search query generator into separate models, and further coordinates them through a dynamic critic-and-generator research cycle.

under-covered topics; and (iii) The disentangled design enables explicit *budget control* at the blueprint level: the critic can independently specify how many documents to retrieve per key point, preventing context-window overload and ensuring balanced evidence coverage across all aspects of the report. We formalize this alternating interaction as a dual-agent cooperative MDP and show that it admits a *bi-level potential game* structure: in the inner game the two agents alternately improve the outline for each query under a shared quality objective, while in the outer game their respective harnesses are alternately optimized across queries toward the same end-to-end quality metric. This two-level structure provides a principled optimization landscape that encourages steady improvement, which we verify empirically through convergence analysis across optimization rounds.

We evaluate AgentDisCo on three widely adopted deep research benchmarks, namely DeepResearch-Bench (Du et al., 2025), DeepConsult (Consult, 2025), and DeepResearchGym (Coelho et al., 2025), using Gemini-2.5-Pro (DeepMind, 2025a) as the base model. AgentDisCo w/ Harness achieves a RACE score of 54.11 on DeepResearchBench and 7.12 on DeepConsult, surpassing leading closed-source systems including Doubao-Research (Research, 2026a), Claude-DeepResearch (Anthropic, 2025), and OpenAI-DeepResearch (OpenAI, 2025a). However, we find that these existing benchmarks predominantly focus on academic or domain-specific consulting queries. To better reflect real-world information needs, we introduce **GALA** (**G**eneral **AI** **L**ife **A**ssistants), a benchmark of authentic, everyday research queries mined from anonymized user interaction histories. Compared with prior benchmarks, the resulting queries exhibit a markedly more lifestyle-oriented character, with dominant topics shifting from Science & Technology and Finance & Business toward everyday domains such as Home & Hobbies, Fashion & Beauty, and Travel. By grounding evaluation in organic user activity while filtering personally identifying information, GALA offers a more faithful reflection of everyday information needs. Alongside the query set, we provide a comprehensive evaluation protocol built upon Gemini-3-Flash (DeepMind, 2025b), in which reports generated by AgentDisCo serve as reference outputs against which competing systems are scored in a pairwise manner. AgentDisCo consistently outperforms strong proprietary baselines on GALA.

We additionally develop a plugin-style rendering agent that can transform unstructured reports into generative UI presentations such as Rednote-style posters, bridging the gap between research outputs and end-user consumption. Building upon this capability, we construct a product demonstration, named AutoResearch Your Interest, which automatically curates and delivers personalized deep research recommendations tailored to individual user profiles. Our demonstration is depicted through Figure 10 in Appendix E.1.

Our main contributions are threefold: (i) **AgentDisCo**, a disentangled and collaborative architecture that separates information exploration from exploitation into a critic agent and a generator agent, mitigating the self-evaluation bias and exploration collapse inherent in monolithic designs and enabling blueprint-level budget control for context management (Figure 1); (ii) **AgentDisCo w/ Harness**, a harness optimization framework for AgentDisCo, formalized as a bi-level potential game, that alternately optimizes the critic and generator scopes and autonomously constructs a policy bank of reusable search strategies (Figure 2); and (iii) **GALA**, a lifestyle-oriented benchmark for everyday deep research needs (Figure 3).

2 A Disentangled and Collaborative Agentic Architecture

2.1 Related Work and Design Philosophy

Open-ended Deep Research Agents. Commercial deep research systems such as GPT Deep Research (OpenAI, 2025a), Gemini Deep Research (Research, 2025), and Kimi-Researcher (Moonshot AI, 2025) have demonstrated the promise of autonomous web-scale information synthesis. Open technical reports including Deep Researcher (Han et al., 2025), WebWeaver (Li et al., 2025), and RhinoInsight (Lei et al., 2025) further reveal architectural patterns such as outline-guided iteration and multi-step retrieval. More broadly, iterative self-refinement methods such as Self-Refine (Madaan et al., 2023) and CRITIC (Gou et al., 2024) show that LLMs can improve outputs through generate-then-critique loops, but rely on the same model for both generation and evaluation, making them susceptible to self-evaluation bias (Panickssery et al., 2024; Park & Choi, 2026). All of the above systems entangle information exploration and exploitation within a single agent.

Formally, these systems can be characterized as a single-agent MDP $\mathcal{M}_{\text{single}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, T \rangle$, where a unified state s_t conflates the search plan with the evolving outline, and the same policy π is responsible for both proposing queries and synthesizing evidence. AgentDisCo factorizes this into a dual-agent disentangled and cooperative MDP $\mathcal{M} = \langle \mathcal{S}^{\mathrm{c}}, \mathcal{S}^{\mathrm{g}}, \mathcal{A}^{\mathrm{c}}, \mathcal{A}^{\mathrm{g}}, \mathcal{P}, \mathcal{R}, T \rangle$, where the exploration state $\mathrm{s}_t^{\mathrm{c}}$ and exploitation state $\mathrm{s}_t^{\mathrm{g}}$ are explicitly separated, each governed by a dedicated policy (π^{c} and π^{g}). This separation enables independent optimization of each role, eliminates the self-evaluation bias inherent in single-agent designs, and allows the critic to provide independent, gap-aware feedback distinct from the generator's own assessment.

Meta-Optimization of Agent Harnesses. Meta-Harness (Lee et al., 2026) introduces end-to-end code-space search over the full agent pipeline, optimizing prompts, tools, and control flow jointly. HarnessBank (Luo et al.) applies MAP-Elites quality-diversity search to maintain a structured archive of harness variants. Both methods optimize offline and freeze the resulting configuration before deployment. In contrast, AgentDisCo's harness conducts joint optimization over the critic and generator agents as a bi-level potential game, where the inner game governs per-query outline refinement and the outer game alternately optimizes the critic and generator harnesses across queries (Section 2.3).

2.2 A Dialectical and Cooperative Critic-Generator Architecture

Our AgentDisCo considers open-ended deep research questions without ground-truth answers. Given such a question q, the agentic system must perform *information exploration* (searching relevant evidence) and *information exploitation* (synthesizing the evidence into a structured report). We argue that iterative optimization should occur on *intermediate outlines* rather than final reports, and we further disentangle exploration and exploitation into two specialized but interacting agents, a *critic agent* π^{c} and a *generator agent* π^{g}, whose dialectical and cooperative interplay drives convergence toward a comprehensive, well-grounded outline.

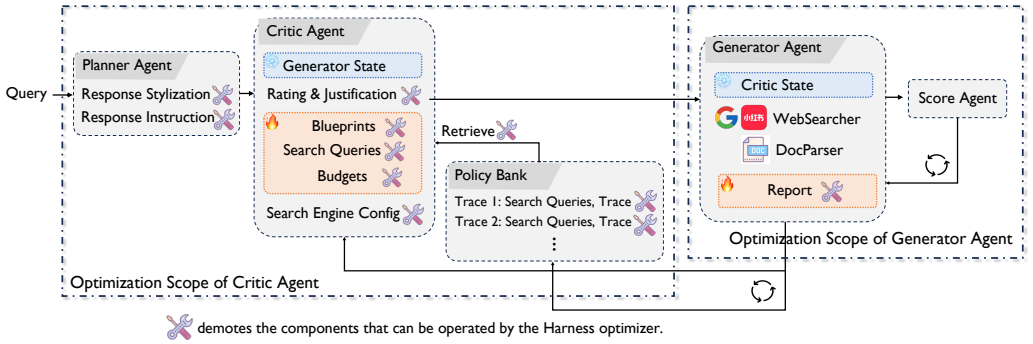

Figure 2: **Overview of the harness optimization in AgentDisCo.** AgentDisCo can automatically discover design strategies by constructing a meta-optimization harness around the dialectical optimization loop. Specifically, the generator agent—originally tasked with producing target outlines—is repurposed as a scoring agent that evaluates the critic agent's outputs and generates quality signals, thereby enabling systematic optimization of the search queries.

We formalize this disentangled and collaborative interaction as a dual-agent cooperative MDP: $\mathcal{M} = \langle \mathcal{S}^{\mathrm{c}}, \mathcal{S}^{\mathrm{g}}, \mathcal{A}^{\mathrm{c}}, \mathcal{A}^{\mathrm{g}}, \mathcal{P}, \mathcal{R}, T \rangle$, where the joint environment state at timestep t is the pair $\mathrm{s}_t = (\mathrm{s}_t^{\mathrm{c}}, \mathrm{s}_t^{\mathrm{g}})$. The *generator state (i.e., the information-exploitation state)*, denoted $\mathrm{s}_t^{\mathrm{g}} = (\mathrm{O}_t, \mathrm{R}_t)$, consists of the current outline O_t and the set of references R_t attached to it. The *critic state (i.e., the information-exploration state)*, denoted $\mathrm{s}_t^{\mathrm{c}} = (\mathrm{B}_t, \mathrm{Q}_t)$, comprises a set of K_t blueprints $\mathrm{B}_t = \{b_{t,1}, \ldots, b_{t,K_t}\}$ together with their associated search queries Q_t. Each blueprint is a triple $b_{t,k} = (p_{t,k},\, q_{t,k},\, n_{t,k})$, where $p_{t,k}$ is a set of key points that the final report should cover, $\mathbf{q}_{t,k} \subseteq \mathrm{Q}_t$ is the corresponding list of targeted search queries dedicated to filling its information gap, and $n_{t,k} \in \mathbb{N}$ is the *retrieval budget* for each key point – the maximum number of retrieved documents needed to gather sufficient information. The disentangled architecture enables the blueprint-level budget control, which can prevent any single key point from monopolizing the context window and ensures the generator receives a balanced evidence set.

Given a user query q, the system initializes the generator state as $\mathrm{s}_0^{\mathrm{g}} = (\varnothing, \varnothing)$, indicating an empty outline and an empty document pool. Conditioned solely on q, the critic agent then performs a coarse decomposition and proposes an initial blueprint set $\mathrm{s}_0^{\mathrm{c}} \sim \pi^{\mathrm{c}}(\cdot \mid q, \varnothing, \varnothing)$, which seeds the subsequent iterative refinement. At each iteration t, the two agents act sequentially rather than simultaneously, with the critic moving first. At the critic's step, conditioned on the question q, the previous generator state $\mathrm{s}_{t-1}^{\mathrm{g}}$ and its own previous state $\mathrm{s}_{t-1}^{\mathrm{c}}$, the critic agent assesses the completeness and quality of the current outline and produces an updated blueprint set: $\mathrm{s}_t^{\mathrm{c}} \sim \pi^{\mathrm{c}}(\cdot \mid q, \mathrm{s}_{t-1}^{\mathrm{g}}, \mathrm{s}_{t-1}^{\mathrm{c}})$. That is, the critic's action is to instantiate the next exploration state. At the generator's step, conditioned on the freshly updated critic state $\mathrm{s}_t^{\mathrm{c}}$ together with its previous state $\mathrm{s}_{t-1}^{\mathrm{g}}$, the generator agent executes the queries prescribed by each blueprint, retrieves new evidence via the search tool, and accordingly revises both the outline and its references: $\mathrm{s}_t^{\mathrm{g}} \sim \pi^{\mathrm{g}}(\cdot \mid q, \mathrm{s}_{t-1}^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}})$. The generator's action thus realizes the exploitation counterpart of the critic's exploration. As for environment dynamics, given the actions of both agents, the joint state transition is deterministic: each agent's output directly instantiates the corresponding component of the next joint state, i.e., $\mathrm{s}_{t+1}^{\mathrm{c}} = \mathrm{a}_t^{\mathrm{c}}$ and $\mathrm{s}_{t+1}^{\mathrm{g}} = \mathrm{a}_t^{\mathrm{g}}$. Consequently, all stochasticity of the trajectory originates from the agents' policies themselves.

Under the above sequential protocol, an interaction trajectory of AgentDisCo unfolds as the alternating sequence $\tau = \{\, q,\, \mathrm{s}_0^{\mathrm{c}},\, \mathrm{s}_0^{\mathrm{g}},\, \mathrm{s}_1^{\mathrm{c}},\, \mathrm{s}_1^{\mathrm{g}},\, \cdots \}$, in which the critic and the generator alternately update their respective state components. The likelihood of sampling τ admits the following equivalent factorizations:

$$
\begin{aligned}
p(\tau) &= \prod_{t=0}^{T-1} \underbrace{\pi^{\mathrm{c}}\big(\mathrm{s}_{t+1}^{\mathrm{c}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}}\big)}_{\text{(1) Critic agent policy}} \underbrace{\pi^{\mathrm{g}}\big(\mathrm{s}_{t+1}^{\mathrm{g}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_{t+1}^{\mathrm{c}}\big)}_{\text{(2) Generator agent policy}} \\
&= \prod_{t=0}^{T-1} \underbrace{\pi^{\mathrm{c}}\big(\mathrm{a}_t^{\mathrm{c}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}}\big)}_{\text{(1) Critic agent policy}} \underbrace{\mathbb{I}\big[\, \mathrm{s}_{t+1}^{\mathrm{c}} = \mathrm{a}_t^{\mathrm{c}} \big]\, \mathcal{P}^{\mathrm{c}}\big(\mathrm{s}_{t+1}^{\mathrm{g}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{a}_t^{\mathrm{c}}\big)}_{\text{(2) Environment dynamic of critic agent}}
\end{aligned}
\tag{1}
$$

Here, T denotes the maximum number of optimization rounds and $\mathbb{I}(\cdot)$ is the indicator function. The first line presents the most compact policy-only factorization, while the latter further decouples each

agent's stochastic decision from its deterministic environment transition, thereby making explicit the loci at which learning signals can be injected.

Although both agents are optimized toward a common objective, their per-round roles are functionally dialectical, jointly forming a cooperative loop. The critic agent π^c challenges the generator's current outline, surfacing missing evidence and uncovered key points through newly proposed blueprints, thereby pushing exploration outward. The generator agent π^g responds by expanding and grounding the outline using the freshly retrieved evidence, thereby consolidating exploitation inward. This dialectical and cooperative interaction progressively refines the outline along the axes of coverage, factual grounding, and structural coherence. Both agents share a common cooperative reward that quantifies the quality of the updated outline for the subsequent writer agent and render agent.

For space limitations, we omit the details of the planner agent, the writer agent, and the render agent in the main text; please refer to Appendix B for details.

2.3 Harness Optimization as A Bi-Level Potential Game

As described in Section 2.2, the core of AgentDisCo lies in the dual optimization between the critic and generator agents. Eq. (1) reveals that, although the two agents assume functionally dialectical roles in each round, they jointly constitute a cooperative loop: both are optimized toward a shared objective, namely the production of a high-quality outline that can attain a sufficiently high score under the critic's evaluation. The resulting artifact, denoted as O_t and R_t, serves as the foundation for the subsequent writing and rendering stages.

Our central insight is that LLMs excel at extracting and summarizing information, but are comparatively weaker at generating effective search queries across heterogeneous retrieval sources. We therefore focus on harness-based optimization for the critic and generator agents, leaving the optimization of the writer and render agents to future work. As illustrated in Figure 2, we introduce a score agent that analyzes the returned search results and provides structured feedback signals to guide iterative optimization. Since the search queries are organized under a set of blueprints, evaluating individual results in isolation is insufficient. We therefore design several criteria – completeness, diversity, search coverage, and internal correlation – and apply them to each search result beyond the final report, aggregating the per-result scores into statistics and distributions over the entire result set. The detailed prompt used by the score agent is provided in Appendix F.2.

Our harness is built by prompting the code agent with only a minimal cue – "you are allowed to store and retrieve traces to evolve" – along with a hint toward bi-level optimization, which can be formulated as the shared potential for the corresponding bi-level potential game:

$$\mathcal{J}(h^{\mathrm{c}}, h^{\mathrm{g}}) = \mathbb{E}_q \, \mathbb{E}_{\tau \sim \prod_{t=0}^{T-1}} \underbrace{\mu^{\mathrm{r}} \, \pi^{\mathrm{c}}\big(\mathrm{a}_t^{\mathrm{c}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}}, \mathrm{m}_t; h^{\mathrm{c}}\big)}_{\text{(1) Critic scope}} \cdot \underbrace{\mu^{\mathrm{w}} \, \pi^{\mathrm{g}}\big(\mathrm{s}_{t+1}^{\mathrm{g}} \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{a}_t^{\mathrm{c}}; h^{\mathrm{g}}\big)}_{\text{(2) Generator scope}} \big[\pi^{\mathrm{s}}(\tau)\big] \tag{2}$$

Here, the code agent implements a simple policy bank under the above prompt. m_t denotes the trace sampled at step t from the policy bank M_t. The retrieval process is formulated as $\mathrm{m}_t \sim \mu^{\mathrm{r}}\big(\cdot \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}}, \mathrm{M}_t\big)$, where μ^{r} denotes the retrieval function. In practice, μ^{r} ranks candidate traces by the product of textual similarity over the entire trace τ (computed via BM25 (Robertson & Zaragoza, 2009)), the quality score assigned by the score agent, and an exponential time-decay factor that favors recently acquired experience. After each step, the newly produced trace is written back into the policy bank, yielding the updated bank $\mathrm{M}_{t+1} \sim \mu^{\mathrm{w}}\big(\cdot \mid q, \mathrm{s}_t^{\mathrm{g}}, \mathrm{s}_t^{\mathrm{c}}, \mathrm{m}_t, \mathrm{a}_t^{\mathrm{c}}, \mathrm{M}_t\big)$, where μ^{w} is a lightweight writing function. Together, μ^{r} and μ^{w} constitute a simple yet effective read-write interface that enables the policy bank to evolve continuously throughout the optimization process.

Eq. (2) reveals the bi-level optimization structure: when optimizing the generator harness h^{g}, the critic harness h^{c} is frozen and the generator is updated to improve outline synthesis and evidence integration; when optimizing the critic harness h^{c}, the generator harness h^{g} is frozen and the critic is updated to improve search-query quality and blueprint coverage. The two stages alternate iteratively, and because both maximize the same global objective $\mathcal{J}(h^{\mathrm{c}}, h^{\mathrm{g}})$, each alternating step is guaranteed to increase the shared potential, encouraging steady improvement – which we verify empirically in Section 4.2.

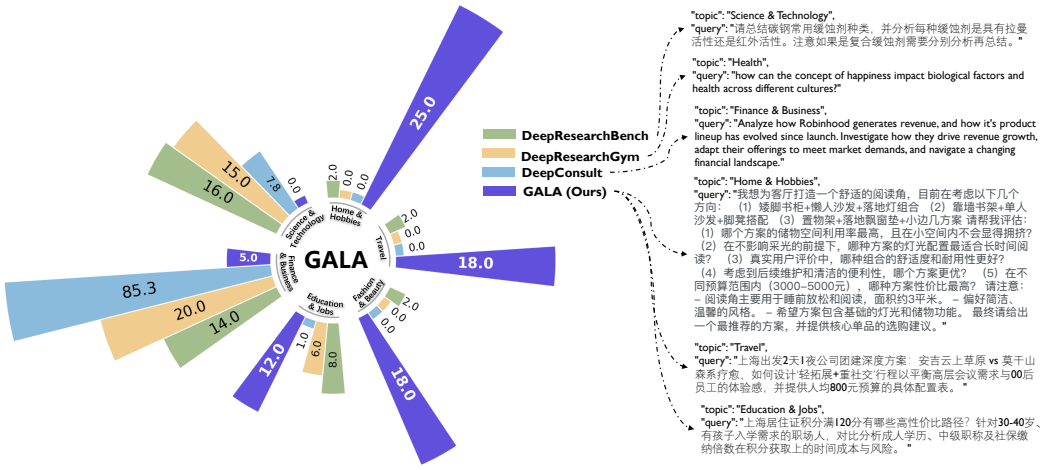

Figure 3: **Comparison between GALA and existing deep research benchmarks.** GALA complements DeepResearchBench, DeepResearchGym, and DeepConsult by emphasizing everyday-life topics that are underrepresented in existing benchmarks.

3 GALA: A Benchmark for General AI Life Assistants

Existing deep research benchmarks (Consult, 2025; Coelho et al., 2025; Du et al., 2025) are typically built from interactions with web-search-enabled LLM chatbots, but their queries remain concentrated in scientific reports, consulting-style analysis, and other professional scenarios. To better capture everyday information needs, we leverage anonymized Rednote interactions from highly active users, including clicks, comments, and browsing records, and use Gemini-3-Flash (DeepMind, 2025b) to infer latent research interests and reformulate them into structured deep research queries. Detailed prompts are provided in Appendix E.1.

We then distill 100 high-quality GALA queries from 260,000 generated candidates through an inspection pipeline that combines automated LLM screening, optional difficulty expansion, and human verification. The automatic review committee, powered by Gemini-3-Pro (DeepMind, 2026a), evaluates each query for naturalness and clarity, reliance on Rednote-specific community knowledge without privacy leakage, and plausibility as a real-world daily-life information need.

4 Experiments

We evaluate AgentDisCo on three public deep research benchmarks, DeepResearchBench (Du et al., 2025), DeepConsult (Consult, 2025), and DeepResearchGym (Coelho et al., 2025), together with our proposed GALA benchmark for general AI life assistants. We follow each benchmark's official evaluation protocol and recommended judge models, including RACE/FACT for DeepResearchBench, pairwise comparison against OpenAI-DeepResearch for DeepConsult, LLM-based quality assessment for DeepResearchGym, and RACE for GALA. We compare against leading deep research systems, including LangChain-Open-Deep-Research (LangChain, Inc., 2023), Doubao-Research (Research, 2026a), Kimi-Research (Moonshot AI, 2025), Claude-Research (Anthropic, 2025), OpenAI-DeepResearch (OpenAI, 2025a), Gemini-2.5-Pro-DeepResearch (Research, 2025), WebWeaver (Li et al., 2025), and TTD-DR (Han et al., 2025), and GALA-specific baselines collected from Doubao-Research, Qwen-Research (Research, 2026c), and OpenAI o3-DeepResearch (Research, 2026b) in April 2026. Public benchmark results for compared systems are taken from prior reports (Li et al., 2025; Han et al., 2025). For RACE-based evaluations, we follow DeepResearchBench (Du et al., 2025) by using an LLM meta-evaluator to allocate query-specific weights over comprehensiveness, insight, instruction-following, and readability, followed by reference-based scoring with AgentDisCo reports provided as anonymized supplementary references. We report FACT scores for DeepResearchBench, where cached page snapshots are available, but omit them for GALA because the metric depends on live web-page availability, which makes citation verification unstable when referenced pages later become inaccessible. Full benchmark descriptions, metrics, judge models, and data-collection details are provided in Appendix C.

Table 1: Performance of agents on DeepResearch Bench in terms of comprehensiveness (Comp.), insight, instruction-following (Inst.), readability (Read.), effective citations (Eff. c.), and citation accuracy (C. acc.). The best results are highlighted with purple color, and the second-best results are highlighted with underlines.

	RACE					FACT	
Agent systems	**Overall**	**Comp.**	**Insight**	**Inst.**	**Read.**	**Eff. c.**	**C. acc.**
Langchain-Open-Deep-Research	43.44	42.97	39.17	48.09	45.22	-	-
Doubao-Research	44.34	44.84	40.56	47.95	44.69	52.62	52.86
Kimi-Research	44.64	44.96	41.97	47.14	45.59	-	-
Claude-Research	45.00	45.34	42.79	47.58	44.66	-	-
Openai-Deepresearch	46.45	46.46	43.73	49.39	47.22	39.79	75.01
Gemini-2.5-Pro-Deepresearch	49.71	49.51	49.45	50.12	50.00	165.34	78.30
WebWeaver (Qwen3-235b-a22b-instruct-2507)	50.62	51.29	51.00	49.98	48.89	**166.73**	78.25
AgentDisCo (Gemini-2.5-Pro)							
w/ Single-agent Architecture	49.77	49.53	50.21	50.12	49.18	55.82	66.73
w/o Retrieval Budget + Harness	51.84	51.62	52.73	51.95	50.98	60.15	78.12
w/o Harness	52.97	52.78	53.89	53.15	51.94	84.94	77.86
w/ Meta-Harness	53.12	52.91	54.03	53.28	52.18	85.47	79.14
w/ Outline-level Harness	53.81	53.56	54.72	53.95	52.87	91.65	78.55
w/ Report-level Harness	53.05	52.83	53.94	53.22	52.13	85.82	79.21
w/ Bi-level Harness	54.11	53.87	55.03	54.25	53.18	92.34	82.12
AgentDisCo (w/o Harness) (Qwen3-235B)	51.82	51.58	52.71	51.93	50.97	111.52	71.83
AgentDisCo (w/o Harness) (Claude-Opus-4.6)	**54.72**	**54.48**	**55.64**	**54.85**	**53.82**	149.88	**83.56**

4.1 Main Results

Results on DeepResearch Bench. As shown in Table 1, AgentDisCo consistently outperforms existing deep-research agent systems on DeepResearch Bench. When using Gemini-2.5-Pro as the backbone model, AgentDisCo achieves an overall RACE score of 52.97, surpassing the prior system based on Gemini-2.5-Pro, i.e., Gemini-2.5-Pro-Deepresearch. The improvement is particularly pronounced in insight, comprehensiveness, and instruction following, where AgentDisCo obtains 53.89, 52.78, 53.15, respectively, compared with 49.45, 49.51, 50.12 from Gemini-2.5-Pro-Deepresearch. In contrast, the gain in readability is relatively moderate, suggesting that the advantage of AgentDisCo does not mainly come from more fluent surface-level writing, but rather from producing more substantive, better-supported, and better-structured research content. We further evaluate the effect of the harness optimization. AgentDisCo w/ Bi-level Harness improves the Gemini-2.5-Pro-based AgentDisCo from 52.97 to 54.11 in overall RACE score, with consistent gains across all RACE dimensions, including comprehensiveness, insight, instruction-following, and readability. AgentDisCo with Claude-Opus-4.6 achieves the best overall performance, reaching 54.72 on RACE. Notably, it obtains a substantial insight score of 55.64, far exceeding all other systems, which suggests that AgentDisCo can effectively leverage stronger reasoning capabilities from advanced LLMs. It also achieves the highest citation accuracy of 83.56, showing that the framework remains highly reliable when scaled to a more capable base model. Overall, these results indicate that AgentDisCo is both effective and scalable: its disentangled yet collaborative critic-generator mechanism improves research depth, evidence reliability, and report quality across different backbone models.

Results on DeepConsult and DeepResearchGym. To examine whether AgentDisCo generalizes beyond our main evaluation setting, we further evaluate it on DeepConsult and DeepResearchGym, as reported in Table 2. Overall, AgentDisCo exhibits strong and robust performance across both benchmarks. With Gemini-2.5-Pro as the backbone, AgentDisCo achieves an overall score of 6.91 on DeepConsult, surpassing the Gemini-2.5-Pro-deepresearch baseline in average score. On DeepResearchGym, it obtains nearly perfect scores in both Depth and Breadth, indicating that the proposed iterative research procedure is effective in expanding both the depth and coverage of the generated reports. As analyzed in the previous subsection, we further evaluate the effect of the harness optimization. Compared with the vanilla Gemini-based AgentDisCo, AgentDisCo w/ Bi-level Harness improves the DeepConsult win rate from 53.26% to 59.44% and the overall score from 6.91 to 7.12. On DeepResearchGym, the harness also raises the overall score from 95.63 to 96.42, yielding improvements in clarity, balance, support, and insightfulness while preserving nearly perfect scores in Depth and Breadth.

Results on GALA. Beyond existing benchmarks, we further construct a lifestyle-oriented deep research benchmark, GALA, as introduced in Section 3. Table 3 reports the evaluation results in

Table 2: Performance of agents on DeepConsult in terms of win rate and average scores and on DeepResearchGym in terms of clarity (Cla.), depth, balance (Bal.), breadth (Brea.), support (Sup.), and insightfulness (Ins.). The best results are highlighted with purple color, and the second-best results are highlighted with underlines.

	DeepConsult				DeepResearchGym						
Agent systems	**Win**	**Tie**	**Lose**	**Overall**	**Cla.**	**Depth**	**Bal.**	**Brea.**	**Sup.**	**Ins.**	**Overall**
Doubao-research	29.95	40.35	29.70	5.42	68.85	93.12	83.96	93.33	84.38	83.12	84.46
Claude-research	25.00	38.89	36.11	4.60	86.67	96.88	84.41	96.56	26.77	90.22	80.25
Openai-deepresearch	0.00	100.00	0.00	5.00	84.90	98.10	89.80	97.40	88.40	89.00	91.27
Gemini-2.5-pro-deepresearch	61.27	31.13	7.60	6.70	90.71	99.90	93.37	99.69	95.00	97.45	96.02
WebWeaver (Qwen3-235b-a22b-instruct-2507)	54.74	28.61	16.67	6.47	89.16	97.58	87.68	96.21	95.26	92.85	93.14
AgentDisCo (Gemini-2.5-Pro)											
w/ Single-agent Architect	38.50	42.85	18.65	6.05	88.45	98.55	90.85	98.80	92.38	90.65	93.28
w/o Retrieval Budget + Harness	48.72	38.85	12.43	6.75	89.72	99.65	92.48	99.80	94.82	92.45	94.82
w/o Harness	53.26	37.50	9.23	6.91	90.50	**100.00**	93.75	**100.00**	96.25	93.75	95.63
w/ Meta-Harness	55.35	35.32	9.33	6.93	90.62	**100.00**	93.95	**100.00**	96.55	94.10	95.87
w/ Outline-level Harness	56.86	32.47	10.67	7.06	**90.98**	**100.00**	94.30	**100.00**	97.73	95.22	96.21
w/ Report-level Harness	55.12	35.55	9.33	6.91	90.48	**100.00**	93.82	**100.00**	96.38	94.02	95.78
w/ Bi-level Harness	59.44	30.23	10.33	7.12	90.88	**100.00**	94.48	**100.00**	97.95	95.22	96.42
AgentDisCo (w/o Harness) (Qwen3-235B)	50.15	39.52	10.33	6.64	90.25	99.85	93.42	99.88	95.72	93.15	95.38
AgentDisCo (w/o Harness) (Claude Opus 4.6)	65.88	22.47	11.65	**7.27**	90.85	**100.00**	**97.85**	**100.00**	**98.93**	**98.66**	**97.54**

terms of comprehensiveness, insight, instruction-following, and readability. In this reference-based evaluation, AgentDisCo, instantiated with Gemini-2.5-Pro, serves as the reference system and is therefore assigned a score of 50.00 across all dimensions. Notably, mainstream deep research systems, including Doubao-Research, Qwen-Research, and OpenAI o3-DeepResearch, obtain lower overall scores than this reference. This result highlights the advantage of the AgentDisCo framework itself: rather than relying solely on the capability of a strong backbone model, AgentDisCo benefits from its structured workflow that combines iterative planning, targeted evidence acquisition, and hierarchical synthesis, which is particularly important for lifestyle-oriented research tasks requiring practical relevance, contextual understanding, and user-aligned recommendations. We next examine the effect of harness optimization. AgentDisCo w/ Bi-level Harness improves the overall score from 50.00 to 50.58, with the most evident gain appearing in insight. This suggests that the harness helps stabilize the execution of the multi-step research workflow and enables the agent to produce more informative and analytically useful responses. We then study the impact of retrieval-source selection. From the table, we can observe that AgentDisCo w/ Rednote achieves a higher overall score of 51.02 and obtains the best readability score among all systems. This indicates that Rednote provides domain-relevant lifestyle content that aligns well with the information needs in GALA, leading to responses that are more natural and accessible for lifestyle-oriented scenarios. Combining domain-specific retrieval with harness optimization yields the strongest performance. AgentDisCo w/ Rednote & Bi-level Harness achieves the best overall score of 52.33 and ranks first in comprehensiveness, insight, and instruction-following. In particular, its insight score reaches 54.10, showing that lifestyle-oriented social content, when integrated through a more reliable execution harness, can substantially improve the practical and contextual value of the generated reports. This demonstrates that the gains from Rednote are further amplified when the agent's research process is better controlled and more consistently executed.

4.2 Analysis and Ablations

We provide a comprehensive analysis of AgentDisCo, including component-level contributions, harness design comparisons, convergence behavior, cross-model generalization, and validation against reward hacking.

Component contributions. Tables 1 and 2 share a common ablation ladder that isolates the contribution of each architectural component. Separating the critic and generator from a single-agent architecture improves RACE from 49.77 to 51.84 (+2.07) and DeepConsult Overall from 6.05 to 6.75, confirming that disentangling exploration and exploitation eliminates self-evaluation bias and broadens search coverage. Adding the critic's adaptive retrieval budget further raises RACE to 52.97 (+1.13) and DeepConsult to 6.91, demonstrating that blueprint-level budget control prevents context-window overload and ensures balanced evidence acquisition (Appendix D.7). The full Bi-level Harness achieves the best results (RACE 54.11, DeepConsult 7.12), with each component contributing meaningful gains.

Table 3: Performance of agents on our proposed GALA benchmark in terms of comprehensiveness (Comp.), insight, instruction-following (Inst.), and readability (Read.). The best results are highlighted in purple, and the second-best results are underlined.

	RACE				
Agent systems	**Overall**	**Comp.**	**Insight**	**Inst.**	**Read.**
Doubao-Research (2026-04)	49.82	50.87	47.42	50.65	50.86
Qwen-Research (2026-04)	46.69	45.38	45.36	47.23	49.56
OpenAI o3-DeepResearch (2026-04)	45.88	45.37	42.88	48.04	47.72
AgentDisCo (Gemini-2.5-Pro)	50.00	50.00	50.00	50.00	50.00
w/ Bi-level Harness (Gemini-2.5-Pro)	50.58	50.41	51.24	50.16	49.85
w/ Rednote (Gemini-2.5-Pro)	<u>51.02</u>	50.88	<u>51.11</u>	<u>51.25</u>	**50.95**
w/ Rednote & Bi-level Harness (Gemini-2.5-Pro)	**52.33**	**52.11**	**54.10**	**51.41**	<u>50.67</u>
w/ Rednote & Google (Gemini-2.5-Pro)	50.95	<u>51.21</u>	50.44	50.78	49.79

Harness design ablations. We conduct ablation studies on DeepResearchBench (50 sampled queries) to compare harness configurations. (1) *Optimization scope*: our targeted search-query optimization achieves 54.11 RACE vs. 53.12 for Meta-Harness-style end-to-end optimization (Lee et al., 2026), a larger gain ($+1.14$ vs. $+0.15$) over the no-optimization baseline despite the same round budget (Appendix D.6). (2) *Optimization target*: outline-level optimization achieves 53.81 vs. 53.05 for report-level, owing to structural flexibility and context efficiency—outlines consume $\sim$2,000 tokens vs. $\sim$15,000 for full reports. (3) *Internal components*: removing the policy bank from the Bi-level Harness reduces RACE from 54.11 to 53.69; further removing the iterative loop drops it to 53.33, confirming that both components are essential (Appendix D.2). Additionally, the critic's adaptive budget allocation—distributing retrieval slots proportionally to coverage gaps—improves RACE by $+0.75$ over fixed uniform allocation, with the gap widening across later rounds (Appendix D.7).

Convergence and scaling. End-to-end scores improve steadily with more outline optimization rounds on both DeepResearchBench and DeepResearchGym, with the most notable gains in comprehensiveness and insight (Figures 6 and 7 in Appendix D.1). For harness optimization, the first 20 rounds contribute stable improvements; beyond round 20, performance saturates within a narrow band, and full trace retention edges out sliding-window expiration by 0.03–0.05 RACE points (Appendix D.8). At the default 3 rounds and 20 harness rounds, AgentDisCo costs $\sim$\$2.87 per query; the harness cost is amortizable since the optimized configuration is reused for future queries, reducing online serving cost to $\sim$\$0.46/query (Appendix D.9).

Cross-model generalization. The harness generalizes across five base models (Claude-Opus-4.6, Gemini-2.5-Pro, Kimi-K2.6, Qwen3-235B, Qwen2.5-72B), with consistent $+1.6\%$ to $+2.3\%$ RACE improvements in a zero-shot transfer setting—the harness is optimized once on Gemini-2.5-Pro and applied to other backbones without further tuning—indicating that the learned search strategies generalize across model families; weaker models benefit proportionally more (Appendix D.3). Tables 1 and 2 further show that instantiating AgentDisCo with Claude-Opus-4.6 achieves the best results across all benchmarks (RACE 54.72, DeepConsult 7.27), demonstrating favorable scaling with stronger backbone models.

Validation. To rule out reward hacking, we verify that (1) the locked final evaluator prevents gaming, (2) $\sim$68% of code modifications target search query guidelines with no tampering, and (3) internal and external metrics improve in tandem across all harness checkpoints (Appendix D.4). Human evaluation confirms metric validity: LLM–human agreement reaches 76.0% (DeepResearchBench) and 74.5% (GALA), and annotators prefer optimized reports 70.0% of the time with Fleiss' $\kappa = 0.58$ (Appendix D.5).

5 Conclusion and Future Work

We present AgentDisCo, a disentangled and collaborative multi-agent architecture designed for open-ended deep research tasks. Extensive experiments demonstrate that this architectural design substantially enhances report comprehensiveness, analytical insight, and evidence support. For future work, it would be worthwhile to further investigate rubric-based training paradigms within the AgentDisCo framework, with the aim of empowering open-source LLMs as viable alternatives to their closed-source counterparts.

AI USE STATEMENT

During the preparation of this manuscript, generative AI tools were used solely for grammar, spelling, and stylistic polish. All conceptual formulations, methodological designs, experimental evaluations, and scientific analyses were conceived, verified, and written by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our research utilizes publicly available, open-source deep research benchmarks and datasets, with no human subjects or private data collection involved. All user interaction data used for the GALA benchmark construction has been anonymized to remove personally identifying information. We anticipate that our work will contribute positively to transparent, trustworthy, and efficient deep research agents.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, the detailed algorithmic formulations, system architecture, and evaluation protocols are documented in Section 2 and Section 4. Full implementation details, prompts, and benchmark specifications are provided in the Appendix, and our complete codebase is included in the supplementary material.

REFERENCES

Anthropic. Meet claude, 2025. URL https://www.anthropic.com/claude.

João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe Jin, Jamie Callan, João Magalhães, Bruno Martins, et al. Deepresearchgym: A free, transparent, and reproducible evaluation sandbox for deep research. *arXiv preprint arXiv:2505.19253*, 2025.

Deep Consult. Deep consult. 2025. URL https://github.com/Su-Sea/ydc-deep-research-evals.

Google DeepMind. Gemini 2.5, 2025a. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Google DeepMind. Gemini-3-flash, 2025b. URL https://deepmind.google/models/gemini/flash/.

Google DeepMind. Gemini 3 pro, 2026a. URL https://deepmind.google/models/gemini/pro/.

Google DeepMind. Gemini 3 pro image, 2026b. URL https://deepmind.google/models/gemini/pro/.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. *arXiv preprint arXiv:2506.11763*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Rujun Han, Yanfei Chen, Zoey CuiZhu, Lesly Miculicich, Guan Sun, Yuanjun Bi, Weiming Wen, Hui Wan, Chunfeng Wen, Solène Maître, George Lee, Vishy Tirumalashetty, Emily Xue, Zizhao Zhang, Salem Haykal, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. Deep researcher with test-time diffusion, 2025. URL https://arxiv.org/abs/2507.16075.

LangChain, Inc. LangChain: Building applications with LLMs through composability, 2023. URL https://python.langchain.com/.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Yu Lei, Shuzheng Si, Wei Wang, Yifei Wu, Gang Chen, Fanchao Qi, and Maosong Sun. Rhinoinsight: Improving deep research through control mechanisms for model behavior and context. *arXiv preprint arXiv:2511.18743*, 2025.

Zijian Li, Xin Guan, Bo Zhang, Shen Huang, Houquan Zhou, Shaopeng Lai, Ming Yan, Yong Jiang, Pengjun Xie, Fei Huang, Jun Zhang, and Jingren Zhou. Webweaver: Structuring web-scale evidence with dynamic outlines for open-ended deep research, 2025. URL https://arxiv.org/abs/2509.13312.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. Harnessbank: Semantic gene-bank search with gated verification for agent-harness self-evolution, 2026. *URL https://arxiv. org/abs/2607.13683*.

Qianli Ma, Siyu Wang, Yilin Chen, Yinhao Tang, Yixiang Yang, Chang Guo, Bingjie Gao, Zhening Xing, Yanan Sun, and Zhipeng Zhang. Human-agent collaborative paper-to-page crafting for under $0.1. In *arXiv preprint arXiv:2510.19600*, 2025. URL https://arxiv.org/abs/2510.19600.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Moonshot AI. Kimi-researcher: End-to-end RL training for emerging agentic capabilities. 2025. URL https://moonshotai.github.io/Kimi-Researcher/.

OpenAI. Deep research system card, 2025a. URL https://cdn.openai.com/deep-research-system-card.pdf.

OpenAI. Gpt-4.1-mini-20250414, 2025b. URL https://openai.com/index/gpt-4-1/.

OpenAI. Gpt-4.1-20250414, 2025c. URL https://openai.com/index/gpt-4-1/.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. Paper2poster: Towards multimodal poster automation from scientific papers. In *arXiv preprint arXiv:2505.21497*, 2025. URL https://arxiv.org/abs/2505.21497.

Arjun Panickssery, Samuel R Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802, 2024.

Hyundoo Park and Byungho Choi. When do agent loops mistake stagnation for progress? self-evaluation bias and externally grounded verification in long-running autonomous llm agent loops. *arXiv preprint arXiv:2607.25152*, 2026.

Doubao Deep Research. Doubao deep research. 2026a. URL https://www.doubao.com/chat/.

Gemini Research. Gemini research. 2025. URL https://gemini.google/overview/deep-research/.

O3 Deep Research. O3 deep research. 2026b. URL https://developers.openai.com/api/docs/models/o3-deep-research.

Qwen Deep Research. Qwen deep research. 2026c. URL https://chat.qwen.ai/?inputFeature=deep_research.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Corby Rosset, Ho-Lam Chung, Guanghui Qin, Ethan C Chau, Zhuo Feng, Ahmed Awadallah, Jennifer Neville, and Nikhil Rao. Researchy questions: A dataset of multi-perspective, decompositional questions for llm web agents. *arXiv preprint arXiv:2402.17896*, 2024.

Tao Sun, Enhao Pan, Zhengkai Yang, Kaixin Sui, Jiajun Shi, Xianfu Cheng, Tongliang Li, Wenhao Huang, Ge Zhang, Jian Yang, and Zhoujun Li. P2p: Automated paper-to-poster generation and fine-grained benchmark. In *arXiv preprint arXiv:2505.17104*, 2025. URL https://arxiv.org/abs/2505.17104.

Zexuan Yan, Jiarui Jin, Yue Ma, Shijian Wang, Jiahui Hu, Wenxiang Jiao, Yuan Lu, and Linfeng Zhang. Glyphbanana: Advancing precise text rendering through agentic workflows. In *arXiv preprint arXiv:2603.12155*, 2026. URL https://arxiv.org/abs/2603.12155.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. Postergen: Aesthetic-aware paper-to-poster generation via multi-agent llms. In *arXiv preprint arXiv:2508.17188*, 2025.

A Limitations and Broader Impact

Limitations. AgentDisCo currently focuses on optimizing search-query generation, while leaving writer-agent optimization and final report polishing largely unchanged. Its performance also depends on the quality and coverage of external search engines and LLM judges. In particular, LLM-based evaluators may exhibit positional bias, verbosity preference, or self-enhancement bias, which can affect the reliability of automatic metrics. To mitigate this, we conduct human evaluation on both DeepResearchBench and GALA, achieving 76.0% and 74.5% LLM–human agreement respectively, and report annotator preference rates with inter-annotator reliability (Fleiss' $\kappa = 0.58$). For GALA, although we anonymize user-interaction patterns and filter personally identifying information, the benchmark remains tied to one platform's content ecosystem and may not fully represent users from other regions or communities. Finally, some live web references can become unavailable over time, making factuality evaluation with URL-fetching metrics difficult to reproduce.

Broader Impact. Deep research agents can improve access to complex information by helping users synthesize evidence across many sources. At the same time, such systems may amplify retrieval bias, produce overconfident reports, or enable persuasive misinformation if misused. We mitigate these risks by requiring source traceability, reporting limitations of live-web evaluation, and treating user-derived benchmark construction as an anonymized and filtered process. Future releases should include clear usage guidelines, privacy documentation, and safeguards for datasets or demos built from user-interaction data.

B Architecture Details

B.1 Planner Agent

To accommodate the heterogeneous nature of deep research queries encountered on a large content platform, we introduce a planner agent that classifies each incoming query into one of two top-level categories, each further subdivided into fine-grained intents: information seeking (including fact query, status & progress, news & information, deep exploration, and resource locating) and decision making (comparison & selection, recommendations & suggestions, how-to guide, travel planning, and purchase decision). Beyond the categorical label, the planner also infers an expected response style tailored to the predicted intent, which subsequently conditions the downstream critic and generator agents. We instantiate the planner with Gemini-2.5-Pro (DeepMind, 2025a) as the backbone, and provide the full prompt template in Appendix E.3. During harness optimization, we grant the planner agent access to generate instructions for the subsequent agentic workflows. Concretely, we observe that the code-generation agent specifies instructions such as tips for formulating search queries and selecting hyperparameters for the search engines, which are then passed along to the downstream outline critic and generator agents.

B.2 Writer Agent

As introduced in Section 2.2, the document bank effectively filters out irrelevant content, thereby alleviating the burden on the model's attentional capacity. Since the generated outline is inherently structured, it is straightforward to partition the outline together with its associated references into a sequence of self-contained chunks, following the strategy of Li et al. (2025). This decomposition reduces the complex task of long-context writing into a series of manageable, attention-focused subtasks, each operating on only the relevant evidence. Because each document has already been assigned a unique index within the outline, the relevant evidence for any given section can be retrieved directly from the document bank. The composition of each section is therefore not a single monolithic action, but rather a deliberate intra-sectional reasoning cycle: at each step, the writer conditions on the previously generated chunks and continues the narrative in a coherent, context-aware manner. This internal monologue is critical for moving beyond shallow summarization toward genuine synthesis across evidence. Finally, the system outputs a Markdown-formatted report in which every cited reference is accompanied by its corresponding URL, ensuring full source traceability. Notably, the writer agent is additionally conditioned on the response style produced by the planner agent, ensuring that the generated report remains coherent with the input query and faithfully aligned with the user's intent.

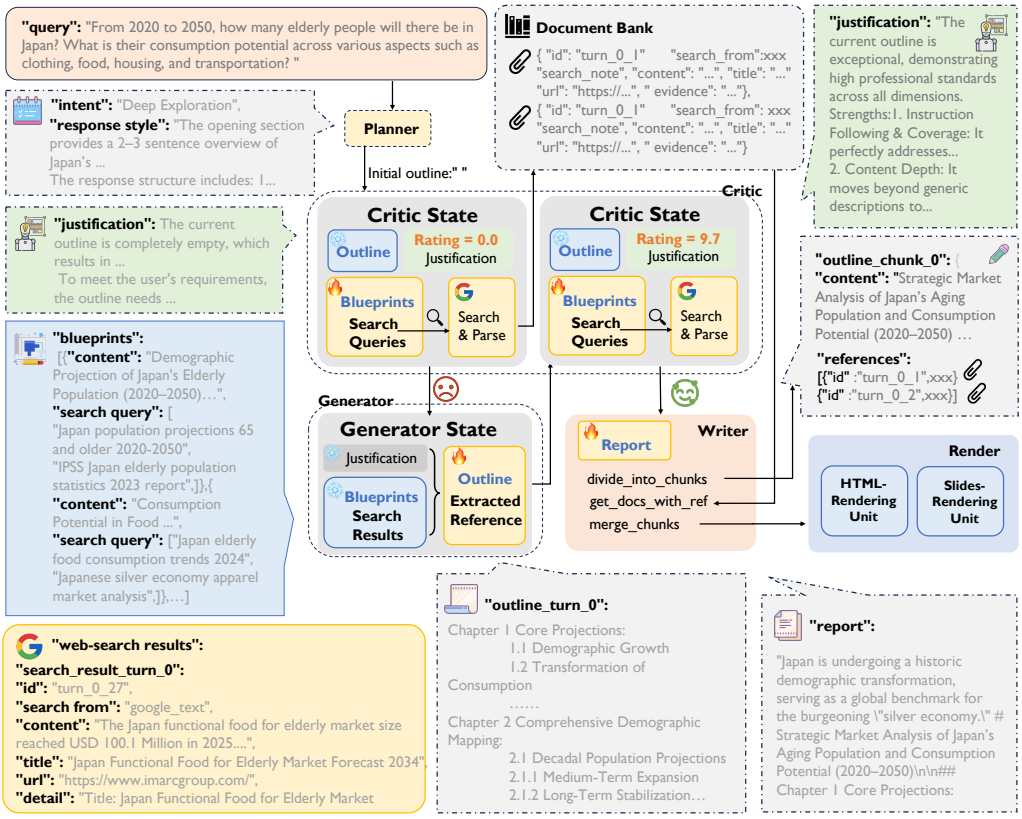

Figure 4: **A showcase of AgentDisCo.** The figure illustrates the end-to-end processing pipeline of an input query: starting from the planner agent, proceeding through the iterative optimization loop between the (outline) critic agent and the (outline) generator agent, and finally passing to the writer agent and the render agent.

As discussed in Section 2.2, our key premise is that extracting and summarizing information does not constitute the primary bottleneck for LLMs, in contrast to the more challenging task of generating effective search queries across heterogeneous retrieval sources. Accordingly, this paper focuses on harness-based optimization for search query generation, leaving the optimization of the writer agent to future work.

B.3 Render Agent

Considering that a clear, intuitive, and visually engaging interface is essential for bridging the gap between raw research outputs and end-user consumption, we develop a render agent that transforms structured research reports into visually rich presentations—ranging from RedNote-style posters to slide decks and HTML pages—thereby allowing users to readily digest, share, and act upon the generated content. Existing open-source render agents (Sun et al., 2025; Zhang et al., 2025; Ma et al., 2025; Pang et al., 2025; Yan et al., 2026) rely heavily on complex, tightly coupled pipelines that target a single output modality, making it difficult to accommodate heterogeneous user preferences over presentation forms. Moreover, their intricate iterative design and rigid workflow orchestration limit the extensibility of packaging multimodal pipelines as reusable tools or skills, while incurring additional deployment and debugging overhead. In contrast, we present a clean and flexible render agent that generalizes across diverse rendering tasks within a unified framework.

As depicted in Figure 5, our render agent incorporates an information extractor to extract key information points from the report. To enhance its performance, we also feed the blueprints and response style into the information extractor as auxiliary inputs. To ensure flexibility in use, we construct a webpage template set $\mathcal{T}^{\mathrm{w}}$ and a slide template set $\mathcal{T}^{\mathrm{s}}$. Given an input document, the information extractor first produces structured multimodal assets conditioned on either $\mathcal{T}^{\mathrm{w}}$ or $\mathcal{T}^{\mathrm{s}}$.

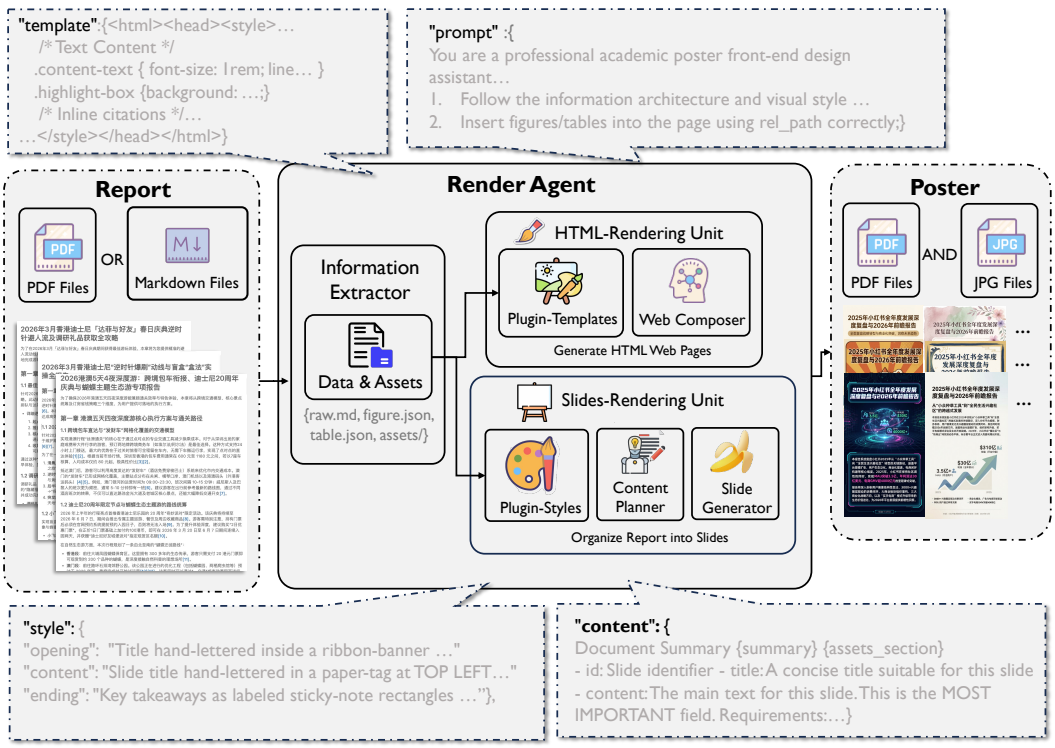

Figure 5: **Overview of the render agent in AgentDisCo.** Our render agent accepts as input a report in either PDF or Markdown format. It first extracts the salient features and structural elements from the report, and then reorganizes the content into one of two presentation modalities: an HTML-based layout or a slide-style layout. Notably, both modalities support pluggable templates and styling components, enabling flexible visual customization. The final output is rendered as a PDF document or a sequence of images, depending on the chosen modality.

We then apply Gemini-2.5-Pro (DeepMind, 2025a) as the web composer to generate webpages in HTML format, or Gemini-3-Pro-Image (DeepMind, 2026b) as the slide generator to produce multiple images for constructing posters or slides. In addition, we provide an option to generate Rednote-style posters, in which the textual content is produced by Gemini-2.5-Pro (DeepMind, 2025a) acting as the content planner. We present multiple showcases of our rendered posters in the Appendix G.1, along with interactive demos.

B.4 A Running Example

To illustrate the end-to-end workflow of AgentDisCo, we present a real execution trace in Figure 4 for the example query: "From 2020 to 2050, how many elderly people will there be in Japan, and what is their consumption potential across various aspects such as clothing, food, housing, and transportation?".

First, the planner agent generates a response style based on the input query, which is then passed, together with the query, into the optimization loop between the outline critic agent and the outline generator agent. At round 0, the critic agent observes an empty outline and assigns a rating of 0. At round 1, the critic agent evaluates the draft and identifies that the housing section is underdeveloped. Accordingly, it emits an updated set of blueprints with refined search queries specifically targeting elderly housing consumption. The generator then revises the outline in response, while the document bank preserves the valid citations accumulated from turn 0, ensuring incremental knowledge accumulation across turns. This trace illustrates how the outline critic and generator agents co-evolve through the exchange of blueprints, rather than one dominating the other. Finally, the outline that receives the highest score is forwarded to the downstream agents: the writer agent composes the report by sequentially elaborating each section of the outline, drawing on the associated references retrieved

from the document bank, and the render agent subsequently transforms the report into the desired presentation format.

C Experiment Setups

C.1 Benchmarks.

We evaluate AgentDisCo on three publicly available benchmarks, together with the proposed GALA benchmark.

- **DeepResearchBench** (Du et al., 2025) comprises 100 PhD-level complex research tasks formulated by domain experts across 22 fields, including Science & Technology, Finance & Business, Software Engineering, and Art & Design.
- **DeepConsult** (Consult, 2025) is a specialized collection of prompts for in-depth research in business and consulting domains, including marketing strategy, financial analysis, emerging technology trends, and business planning.
- **DeepResearchGym** (Coelho et al., 2025) evaluates performance on real-world complex queries. It contains 100 queries sampled from the large-scale Researchy Questions dataset (Rosset et al., 2024), which comprises approximately 96,000 authentic information-seeking queries.
- **GALA** is a deep research benchmark for general AI life assistants, introduced in Section 3.

C.2 Metrics.

We adopt the official evaluation metrics and recommended judge LLMs for each benchmark.

- **DeepResearchBench** (Du et al., 2025) employs RACE (Report Quality) and FACT (Web Retrieval via Google Search). RACE evaluates Comprehensiveness (Comp.), Insight/Depth (Insight), Instruction-Following (Inst.), and Readability (Read.), with an overall score computed as a weighted sum. FACT measures Citation Accuracy (C. Acc.) and Average Effective Citations per Task (Eff. c.). Following the benchmark protocol against Gemini-2.5-Pro-DeepResearch, we adopt Gemini-2.5-Pro (DeepMind, 2025a) as the judge model.
- **DeepConsult** (Consult, 2025) evaluates performance via pairwise comparison against the OpenAI-DeepResearch baseline. The primary metrics are win rate, tie rate, loss rate, and average quality score. The judge model is GPT-4.1-20250414 (OpenAI, 2025c).
- **DeepResearchGym** (Coelho et al., 2025) employs an LLM judge to assess clarity, insightfulness, depth, balance, breadth, support, and overall quality. The judge model is GPT-4.1-mini-20250414 (OpenAI, 2025b).
- **GALA** evaluates report quality using RACE with Gemini-3-Flash (DeepMind, 2025b) as the judge model.

C.3 Statistical Reliability

To ensure statistical reliability, all AgentDisCo experiments on the ablation set (50 sampled queries from DeepResearchBench) are repeated across 3 independent runs with different random seeds, and we report the mean scores. The observed standard deviations are below 0.3 RACE points across all configurations, confirming the stability of our results. Full-benchmark evaluations (100 queries) are conducted with a single run due to computational cost, but are complemented by human evaluation to validate metric reliability.

C.4 Compared Systems

We benchmark AgentDisCo against a suite of leading deep research agents: **LangChain-Open-Deep-Research** (LangChain, Inc., 2023), **Doubao-Research** (Research, 2026a), **Kimi-Research** (Moonshot AI, 2025), **Claude-Research** (Anthropic, 2025), **OpenAI-DeepResearch** (OpenAI, 2025a), **Gemini-2.5-Pro-DeepResearch** (Research, 2025), **WebWeaver** (Li et al., 2025), and **TTD-DR** (Han et al., 2025). Their results on the three public benchmarks are taken directly from Li et al. (2025); Han et al. (2025). For GALA, human annotators collected reports from the official web interfaces of

Doubao-Research (Research, 2026a) and **Qwen-Research** (Research, 2026c), as well as outputs from **OpenAI o3-DeepResearch** (Research, 2026b) obtained via its API, with all data acquired in **April 2026**. Our evaluations adopt AgentDisCo (Gemini-2.5-Pro) as the reference report. We also include a variant of AgentDisCo that uses the Rednote search engine as its retrieval source, denoted as **AgentDisCo w/ Rednote**.

C.5 Statistics and Comparisons with Existing Benchmarks

GALA consists of 100 carefully curated deep research queries. Following the taxonomy proposed in DeepResearchBench (Du et al., 2025), each query is categorized into one of 22 topics: "Finance & Business", "Science & Technology", "Software Development", "Education & Job", "Health", "Literature", "History", "Hardware", "Industrial", "Art & Design", "Games", "Crime & Law", "Entertainment", "Sports & Fitness", "Software", "Transportation", "Religion", "Home & Hobbies", "Travel", "Food & Dining", "Fashion & Beauty", and "Social Life". We adopt Gemini-3-Pro (DeepMind, 2026a) as the query classifier; the detailed prompts are provided in Appendix E.2. The comparison reveals a clear divergence in topical focus across benchmarks. Existing benchmarks are heavily skewed toward professional and technical domains: DeepResearchBench (Du et al., 2025) is dominated by "Science & Technology" (26.8%), "Finance & Business" (17.0%), and "Education & Jobs" (10.0%); DeepConsult (Consult, 2025) is overwhelmingly concentrated in "Finance & Business" (85.3%), followed by "Science & Technology" (7.8%); and DeepResearchGym (Coelho et al., 2025) is led by "Finance & Business" (20.0%), "Science & Technology" (15.0%), "History" (14.0%), "Social Life" (13.0%), and "Health" (11.0%). In contrast, GALA exhibits a markedly different distribution centered on everyday-life domains, with "Home & Hobbies" (25.0%), "Travel" (18.0%), "Fashion & Beauty" (18.0%), and "Education & Jobs" (12.0%) as its dominant categories. This contrast highlights GALA's unique role in complementing existing benchmarks: rather than re-emphasizing professional research scenarios, GALA targets authentic, daily-life information needs that have been largely underrepresented in prior evaluations.

For clarity, Figure 3 illustrates the top six taxonomic categories of GALA along with their corresponding distributions across DeepResearchBench, DeepConsult, and DeepResearchGym. As can be observed, GALA offers essential coverage for the "Home & Hobbies", "Travel", and "Fashion & Beauty" domains that are notably underrepresented in the aforementioned datasets.

C.6 Evaluation Protocols

One prevailing challenge in evaluating open-ended deep research reports is the absence of an exact ground truth for each query. To address this, we adopt the RACE metric proposed in DeepResearchBench (Du et al., 2025). The evaluation proceeds in two stages: (i) **dynamic dimension weight allocation**: an LLM (Gemini-3-Flash (DeepMind, 2025b) in practice), acting as a meta-evaluator, analyzes the input query to determine the relative importance of four evaluation dimensions: Comprehensiveness, Insight, Instruction-Following, and Readability; and (ii) **reference-based pairwise scoring**: a separate LLM (Gemini-3-Flash in practice) scores the target report against criteria derived for each dimension. The final score is computed as a weighted summation over the per-dimension scores. Since RACE evaluation requires a reference report to produce pairwise scores, we provide reports generated by AgentDisCo as corresponding reference reports in anonymized supplementary materials.

Following the DeepResearchBench protocol (Du et al., 2025), the RACE scorer evaluates each article on a 0–10 scale across four dimensions, each decomposed into weighted sub-criteria:

- **Comprehensiveness** (4 sub-criteria, equal weight): Information Coverage Breadth, Information Depth and Detail, Data and Factual Support, Multiple Perspectives and Balance.
- **Insight** (4 sub-criteria, equal weight): Analysis Depth and Originality, Logical Reasoning and Causal Relationships, Problem Insight and Solutions, Forward-Looking and Inspirational Thinking.
- **Instruction Following** (3 sub-criteria, weights 0.34/0.33/0.33): Response to Task Objectives, Adherence to Scope Limitations, Complete Coverage of Task Requirements.
- **Readability** (4 sub-criteria, equal weight): Clear Structure and Logic, Language Expression and Fluency, Appropriate Use of Technical Terms, Information Presentation and Visual Elements.

The scoring rubric maps 0–2 (very poor), 2–4 (poor), 4–6 (average), 6–8 (good), 8–10 (excellent). The final RACE score is a weighted sum of dimension scores, with per-query dimension weights dynamically allocated by the meta-evaluator.

We report FACT scores for DeepResearchBench, where cached page snapshots are available, but do not adopt FACT for GALA evaluation. FACT verifies the factual accuracy of references by fetching their corresponding web content; however, as time passes, many referenced web pages become inaccessible (e.g., returning 404 errors), making the metric unreliable for benchmarks without archived snapshots.

D Additional Experiments and Ablations

D.1 Analysis

Statistics of Outline Optimizations. One of the core ideas of AgentDisCo is its disentangled yet collaborative framework, in which different agents are assigned specialized roles and interact through an iterative optimization process. To directly assess whether this design improves the quality of the generated outline, we evaluate the effect of our outline optimization module in isolation. Since the optimization process is initiated by the critic agents after an initial outline has been produced, at least two rounds are required to observe the effect of critic-guided refinement. To isolate and quantify the contribution of outline optimization, we conduct an ablation study on the end-to-end benchmarks, as reported in Figures 6 and 7. Specifically, we collect samples from DeepResearch Bench and DeepResearchGym and apply up to three rounds of outline optimization, while keeping the subsequent writing strategy unchanged across all settings. This design ensures that performance differences can be primarily attributed to the quality of the optimized outlines rather than variations in the writing process. The benefits of iterative refinement are consistent across both benchmarks. On DeepResearch Bench, the overall score increases steadily as the number of optimization rounds grows, with particularly notable improvements in comprehensiveness and insight. This supports our hypothesis that each optimization round enables the planner to construct a more detailed, coherent, and logically organized outline. A similar trend is observed on DeepResearchGym, where later optimization rounds achieve substantially stronger scores in depth and breadth, indicating more exhaustive coverage of the target topic.

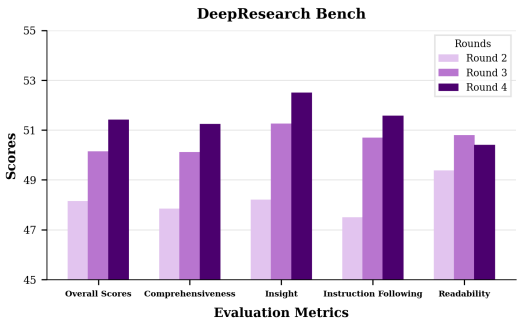

Figure 6: End-to-end scores with varying rounds of outline optimization on DeepResearch Bench.

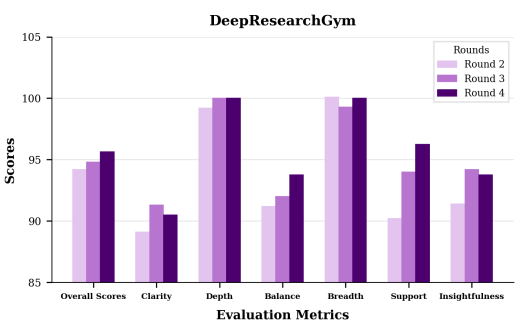

Figure 7: End-to-end scores with varying rounds of outline optimization on DeepResearchGym.

Consistency between Harness-Based Critic Optimization and End-to-End Performance. To evaluate the effectiveness of the proposed harness, we examine whether improvements measured by the harness are consistent with end-to-end performance gains. In the scoring agent of our harness framework, whose detailed prompts are provided in Appendix F.2, we introduce an intermediate metric, named Search Coverage, ranging from 0 to 100, to quantify the quality and coverage of the generated search queries. A higher Search Coverage score indicates that the queries are more likely to capture the key aspects required for answering the research question.

Specifically, we evaluate the Search Coverage scores at optimization rounds 0, 10, and 20, and compare them with the corresponding end-to-end scores on DeepResearch Bench. To reduce evaluation cost while maintaining a representative assessment, we randomly sample 50 examples from DeepResearch Bench as the evaluation pool. As shown in Figure 8, Search Coverage increases from 6.25 at

round 0 to 7.93 at round 10 and further to 8.21 at round 20. This trend is accompanied by a consistent improvement in the end-to-end overall score, which increases from 52.97 to 53.54 and then to 54.11. These results indicate that the harness-based optimization signal is well aligned with downstream benchmark performance. In particular, improving the coverage and quality of search queries leads to more effective evidence acquisition, which in turn contributes to better final reports. Therefore, the proposed harness provides a meaningful and practical intermediate optimization objective for improving end-to-end deep research performance.

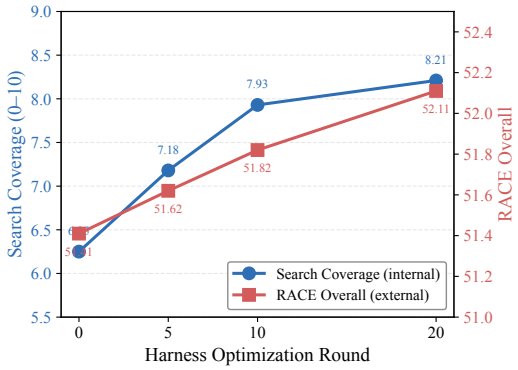

Figure 8: Consistency of harness optimization over the critic agent with end-to-end performance across varying rounds on DeepResearch Bench.

Design Rationale of Bi-Level Optimization. The bi-level potential game formalization (Eq. 2) directly motivates three design choices. First, the bi-level structure prescribes alternating rather than joint optimization: freezing one harness while updating the other avoids co-adaptation instabilities that arise when both scopes change simultaneously. Second, the shared potential guarantees monotonic improvement under alternation, ruling out oscillatory dynamics between the critic and generator—a property empirically confirmed by the steady convergence shown in Figure 8. Third, the policy bank serves as a persistent memory that accumulates the potential gains from prior rounds, enabling experience reuse across queries rather than optimizing each query from scratch.

D.2 Harness-Internal Ablation

To isolate the contribution of each harness component, we conduct an ablation study on DeepResearchBench using 50 randomly sampled queries. We progressively remove components from the full harness and measure the impact on both the internal search coverage metric and the end-to-end RACE score.

Table 4: Harness-internal ablation on DeepResearchBench (50 sampled queries).

Harness Configuration	search_coverage	RACE Overall
Full Harness	8.21	54.11
− Policy Bank (no trace reuse)	78.41	53.69
− Iterative Loop (single-round optimization only)	68.72	53.33
No Harness	6.25	52.97

The Full Harness and No Harness values are consistent with Appendix D.1 (50 sampled queries). The iterative optimization loop contributes the largest gain (+0.78 RACE points), followed by the policy bank (+0.42). This confirms that both components are essential: the iterative loop provides the primary optimization signal, while the policy bank amplifies gains by enabling cross-query knowledge transfer.

D.3 Cross-Model Harness Generalization

To verify that the harness optimization framework is not specific to a single backbone model, we apply it to five base models spanning both frontier and smaller-scale architectures. All experiments use 20 rounds of harness optimization on DeepResearchBench (50 sampled queries).

Table 5: Cross-model harness generalization on DeepResearchBench (RACE Overall).

Base Model	Without Harness	With Harness (20 rounds)	Relative Gain
Claude-Opus-4.6	54.72	55.60	+1.6%
Gemini-2.5-Pro	52.97	54.11	+2.2%
Kimi-K2.6	51.15	52.04	+1.7%
Qwen3-235B	51.82	52.75	+1.8%
Qwen2.5-72B	46.51	47.58	+2.3%

The harness provides consistent, model-agnostic improvements (+1.6% to +2.3%) across all five models, confirming that the optimization framework generalizes beyond a single backbone. Weaker models benefit proportionally more (Qwen2.5-72B: +2.3%, Gemini-2.5-Pro: +2.2% vs. Claude: +1.6%), which is expected: weaker models have more room for improvement in search query generation, and the harness's iterative refinement compensates for their limitations. The harness is effective even for Qwen2.5-72B (72B parameters), demonstrating applicability beyond frontier-scale models.

D.4 Reward Hacking Validation

A key concern is whether the harness optimization genuinely improves research quality or merely games the internal search coverage metric. We provide three lines of evidence against reward hacking.

1) Architectural constraint—the locked final evaluator prevents intermediate gaming. The locked evaluator `JudgeSearchQueryDiversityAPI` receives *both* relevance scores and snippet text from `SummaryQA`, and independently checks whether snippets across different search queries are *informationally complementary*—not just individually relevant. This means score inflation alone cannot game the evaluator, and snippet fabrication would need to fool a separate LLM judge reading the full text against the original search queries.

2) Empirical audit—no tampering observed across 20 optimization rounds. We reviewed all modifications made by the code-generation agent across optimization runs. On average, the distribution of modifications is as follows:

Table 6: Distribution of code modifications during harness optimization.

Modification Category	Avg. Proportion
Search query generation guidelines (outline template)	∼68%
Blueprint update strategy	∼26%
SummaryQA scoring criteria adjustment	∼4%
SummaryQA snippet formatting	∼2%

Across all optimization logs, we found no instance of reward hacking. The small fraction (∼6%) of modifications that touched `SummaryQA` were all minor adjustments (e.g., tightening relevance thresholds to make scoring *stricter*, adjusting snippet length). None attempted to fabricate content or inflate scores. The optimizer consistently focused on improving search query generation strategies—exactly the intended optimization target.

3) Independent external validation—gains confirmed beyond the internal metric. The definitive test is whether internal search coverage improvements translate to gains on the completely independent RACE evaluation (different LLM judge, different criteria, no access to `SummaryQA` outputs):

Table 7: Internal vs. external metric correlation across harness rounds.

Harness Round	search_coverage (internal)	RACE Overall (external)	Relative Gain
0 (baseline)	6.25	52.97	—
5	7.18	53.21	+0.45%
10	7.93	53.54	+1.08%
20	8.21	54.11	+2.15%

Both internal and external scores improve consistently across all four checkpoints (rounds $0 \rightarrow 5 \rightarrow 10 \rightarrow 20$), providing evidence against reward hacking: if the optimizer were gaming `SummaryQA`, the internal score would rise while the external score stagnates or diverges.

D.5 Human Evaluation

1) Cross-benchmark human–LLM agreement. For benchmarks using RACE evaluation (DeepResearchBench and GALA), three annotators performed blind pairwise comparisons ("which report better answers the query?"). We report the agreement between the LLM judge and human majority vote, compared against inter-annotator agreement:

Table 8: Human–LLM agreement on RACE evaluation.

Benchmark	# Pairs	Human–LLM Agreement	Human–Human Agreement
DeepResearchBench	50	76.0%	79.7%
GALA	40	74.5%	78.2%

The LLM judge's agreement with human majority is consistently within 4 percentage points of inter-annotator agreement, confirming that the RACE metrics capture genuine quality differences. The system-level rankings produced by human annotators are fully consistent with those reported in Tables 1 and 3 of the paper. For DeepConsult (pairwise win/tie/lose comparison, judged by GPT-4.1) and DeepResearchGym (multi-dimension quality scoring, judged by GPT-4.1-mini), we follow each benchmark's official evaluation protocol. Human spot-checks on sampled outputs confirm that the automatic rankings are consistent with perceived quality.

2) Human validation of harness optimization (round 0 vs. round 20). To directly verify that the harness produces genuine quality improvement (not just metric gaming), 3 annotators compared 40 query-report pairs from harness round 0 vs. round 20 on DeepResearchBench (blind to round number). The optimized reports were preferred in **70.0%** of cases (28/40), with Fleiss' $\kappa = 0.58$ (moderate agreement). This confirms that the harness scoring signals translate to perceived quality improvement, not just metric gaming.

D.6 Comparison with Meta-Harness

We conduct a controlled comparison by replacing our harness with a Meta-Harness-style end-to-end optimization (Lee et al., 2026) (optimizing the full agent pipeline rather than targeting the search query generation bottleneck).

Table 9: Targeted (AgentDisCo) vs. end-to-end (Meta-Harness-style) optimization.

Optimization Scope	DeepResearchBench (RACE)	DeepResearchGym (Overall)
No optimization (baseline)	52.97	95.63
End-to-end pipeline (Meta-Harness-style)	53.12	95.87
Search query generation only (Ours)	54.11	96.42

While Meta-Harness improves over the baseline, its end-to-end search space dilutes the optimization signal across the full pipeline. By concentrating optimization on the search query generation

bottleneck, AgentDisCo achieves a larger gain (+1.14 vs. +0.15 RACE) with the same round budget. The 0.99-point RACE gap confirms that targeted optimization of the harder sub-task (search query generation) outperforms unfocused end-to-end optimization.

D.7 Adaptive Budget Ablation

A key feature of AgentDisCo's disentangled architecture is blueprint-level budget control: the critic adaptively allocates the per-key-point retrieval budget $n_{t,k}$ based on coverage gaps, subject to the constraint $\sum_{k=1}^{K_t} |\mathbf{q}_{t,k}| \cdot n_{t,k} \leq N_{\max}$. To evaluate the effectiveness of this mechanism, we compare adaptive allocation against a fixed uniform baseline that distributes the same total budget equally across all key points.

Table 10: Adaptive vs. fixed budget allocation across critic–generator iteration rounds on DeepResearchBench (50 sampled queries).

Round	RACE Overall		Search Coverage	
	w/ Adaptive	w/o Adaptive	w/ Adaptive	w/o Adaptive
1	52.21	51.98	5.82	5.60
2	52.55	52.24	6.75	6.21
3 (default)	52.97	52.51	7.41	6.63
4	53.05	52.58	7.58	6.72

As shown in Table 10 and Figure 9, adaptive budget allocation consistently outperforms fixed allocation across all iteration rounds. The gap widens with more rounds (ΔRACE = 0.23 at Round 1 $\rightarrow$ 0.46 at Round 3), confirming that adaptive allocation becomes increasingly beneficial as the critic identifies and targets specific coverage gaps in later iterations. Search coverage follows the same trend, with adaptive allocation achieving 7.41 vs. 6.63 at Round 3.

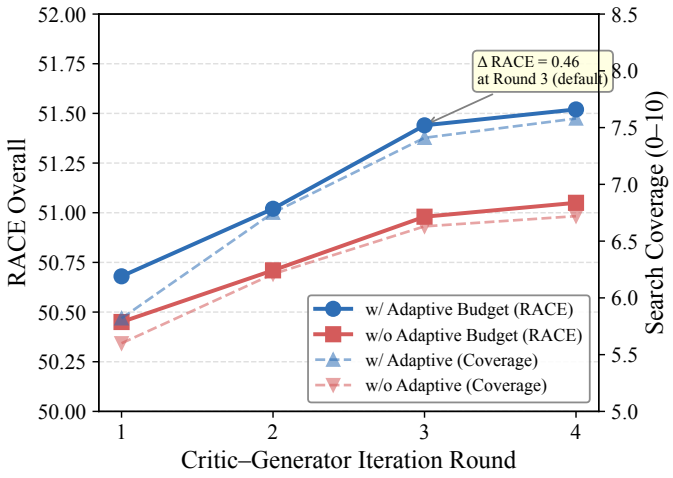

Figure 9: Adaptive vs. fixed budget allocation across critic–generator iteration rounds. Solid lines show RACE Overall (left axis); dashed lines show search coverage (right axis, 0–10 scale). Adaptive allocation yields larger gains in later rounds as the critic focuses retrieval on under-covered key points.

D.8 Extended-Round Policy Bank Experiments

To explore scaling beyond 20 rounds, we introduce a sliding-window expiration mechanism: the policy bank retains only the most recent 2,000 traces, with older traces automatically expired. We compare full retention (keep all traces) and a sliding window (recent 2,000) on DeepResearchBench (50 sampled queries):

Table 11: Extended-round policy bank experiments on DeepResearchBench.

Harness Rounds	Policy Bank Strategy	Bank Size	search_coverage	RACE Overall	Relative Gain
0 (baseline)	—	0	6.25	52.97	—
10	Full (no expiry)	∼1,000	7.93	53.54	+1.08%
20	Full (no expiry)	∼2,000	8.21	54.11	+2.15%
30	Full (no expiry)	∼3,000	8.32	54.05	+2.04%
30	Sliding window (recent 2k)	∼2,000	8.25	54.02	+1.98%
40	Full (no expiry)	∼4,000	8.38	54.19	+2.30%
40	Sliding window (recent 2k)	∼2,000	8.31	54.14	+2.21%
50	Full (no expiry)	∼5,000	8.41	54.15	+2.23%
50	Sliding window (recent 2k)	∼2,000	8.35	54.10	+2.13%

The first 20 rounds contribute a stable +2.15% improvement. Beyond round 20, performance fluctuates within a narrow band (54.0 to 54.2) rather than continuing to climb monotonically. This variance is expected: later rounds explore increasingly marginal search strategies, and the 50-query evaluation set introduces sampling noise at this scale. Across all extended rounds, full retention edges out the sliding window by 0.03–0.05 points on average. Retaining all traces provides a marginal advantage because early-round traces occasionally contain diverse search strategies that complement later, more refined ones. Therefore, 20 rounds with full retention (∼2,000 traces) is the recommended default.

D.9 Cost Analysis

We provide a detailed cost analysis of AgentDisCo, broken down by component:

Table 12: Per-query cost breakdown of AgentDisCo.

Component	Avg. Tokens (Input)	Avg. Tokens (Output)	Avg. Cost ($)
Planner	1,200	350	0.005
Critic Agent (per round)	8,500	2,800	0.038
Generator Agent (per round)	12,300	4,500	0.058
Search API calls (per round)	—	—	0.020
Writer Agent	15,600	8,200	0.102
Total (3 rounds, no Harness)	—	—	**∼$0.46**
Total (3 rounds, w/ Harness 20 rounds)	—	—	**∼$2.87**

AgentDisCo (as described in Section 2.2) uses the critic agent's score to determine early termination—if the outline score exceeds the threshold, optimization stops before the maximum rounds. We also have minimum/maximum round bounds for the agent loop, and harness convergence stop (optimization halts after 5 consecutive rounds without improvement). The cost of ∼$0.46 per query (without harness) is comparable to GPT Deep Research (∼$0.50 per query as estimated from API pricing) and substantially lower than systems requiring human-in-the-loop refinement. The harness optimization cost (∼$2.87) is a one-time amortizable cost: the optimized prompts and code are reused for future queries, so the online serving cost remains about $0.46/query.

D.10 GALA Retrieval Source Ablation

To demonstrate that the GALA benchmark is not inherently tied to Rednote as a retrieval source, we provide the following retrieval-source ablation on GALA (consistent with Table 3, using RACE evaluation with AgentDisCo as reference):

The results show that: (1) AgentDisCo with Google Search alone serves as the RACE evaluation reference (anchored at 50.00), demonstrating that the GALA benchmark is not inherently tied to Rednote as a retrieval source; (2) Rednote provides a meaningful edge for lifestyle queries (+1.02 Overall), which is expected given its community-generated content closely aligned with GALA's everyday information needs; and (3) combining both retrieval sources (50.95) performs comparably

Table 13: Retrieval-source ablation on GALA.

Retrieval Source	AgentDisCo Overall	Comp.	Insight	Inst.	Read.
Google Search only (reference)	50.00	50.00	50.00	50.00	50.00
Rednote only	51.02	50.88	51.11	51.25	50.95
Rednote + Google	50.95	51.21	50.44	50.78	49.79

to Rednote alone (51.02), confirming that the gains are driven by domain relevance rather than search volume. Additionally, the GALA queries themselves are reformulated by Gemini-3-Flash into standard deep research queries—they do not contain Rednote-specific jargon or references. Any web search engine can retrieve relevant information for these everyday-life questions.

D.11 GALA Reference Score Sensitivity

The 50.0 score is a property of the RACE evaluation protocol, not a score we assign. RACE uses reference-based pairwise scoring: one system's output serves as the reference (anchored at 50.0), and all others are scored relative to it. To rigorously verify that the reference choice does not affect relative rankings, we perform a full A↔B swap experiment on GALA: we re-ran the evaluation using each of three different systems as the reference anchor, producing three independent scoring runs:

Table 14: Reference score sensitivity analysis on GALA.

Reference System	AgentDisCo	Doubao	Qwen-Research	OpenAI o3-DR
AgentDisCo = 50.00 (original)	50.00	49.82	46.69	45.88
Doubao = 50.00	50.19	50.00	46.88	46.07
OpenAI o3-DR = 50.00	54.15	53.96	50.83	50.00

The relative ordering (AgentDisCo > Doubao > Qwen-Research > OpenAI o3-DR) and pairwise score gaps are preserved across all three reference choices.

E Prompt Design

E.1 Prompt for Deep Research Query Miner

As depicted in Figure 10, the query miner is designed to elicit users' latent interests by systematically analyzing their browsing history. The detailed prompts are listed below.

Prompt: Research Query Generator

Role Definition
You are a **user behavior analysis + research task design** expert.
Your task is: based on the user's Xiaohongshu (RedNote) browsing history, generate 4–6 complex Deep Research queries.
These queries will be submitted to a Research Agent with the following capabilities:

- Can search for information online
- Can access multiple data sources
- Can perform multi-step reasoning
- Can synthesize multiple information sources to draw conclusions

Therefore, queries must:

- Be clear, structured, and actionable
- Focus on decision-making actions, driving users to make specific decisions
- Require support from multiple information sources and cannot be answered directly by common sense



Figure 10: **Overview of the architecture and applications of AgentDisCo.** AgentDisCo spans the full pipeline from mining latent deep research queries in user interaction histories to producing structured reports and rendering visually rich posters. This end-to-end design realizes the vision of "AutoResearch Your Interest", automatically tracking evolving user interests and delivering personalized deep research recommendations tailored to individual user profiles.

Step 1 — User Interest Modeling *(internal reasoning, do not output)*
Extract three layers of interests from the Xiaohongshu browsing history:
[Long-term Preferences]
Identify ≥3 sustained interest directions, based on:

- High-frequency likes / saves / time-on-post over 60 seconds
- Repeated appearances across different time periods
- Involving multiple brands or content formats

Common directions include: fashion & styling / beauty & skincare / home renovation / fitness & sports / food exploration / travel

[Short-term Interests]
Identify ≥3 recent concentrated browsing hotspots, based on:

- Highly concentrated recent browsing
- High frequency of appearance, high content similarity
- Pointing to a specific product / brand / event / scenario

[Potential Decision Needs]
Based on long-term preferences and short-term interests, infer actual decisions the user may currently face, such as:

- Whether to purchase a certain product
- How to choose among several options
- How to plan a trip or renovation
- Whether to try a new brand or new solution

When generating queries, must ensure:

- At least 2 queries come from long-term preferences
- At least 1 query comes from short-term interests
- Different queries cover different decision types

Step 2 — Research Query Structure *(internal reasoning, do not output)*
Each query must contain the following five components during construction, and be presented in natural language at the end:

[1] Background Context (1–2 sentences)
Describe a real user scenario, clearly stating the user's goal or confusion, such as:

- I'm planning to purchase ...
- I'm planning to ...
- I'm considering whether to ...
- I'm torn between the following options ...

[2] Research Subjects
Clearly list the specific subjects to be researched, using numbered format:

(1) ...
(2) ...
(3) ...

Research subjects can be: products / brands / solutions / locations / platforms / strategies

[3] Research Questions (3–5)
Use numbered format to list questions that require research to answer, such as:

1. What are real user reviews and reputation like
2. Does actual experience match the advertised claims
3. How is the price and value for money
4. What are common issues or usage risks
5. How high is the long-term usage cost or maintenance difficulty

Research questions must be: specific, searchable, comparable, and analyzable

[4] Constraints (2–4)
Clearly state the user's limiting conditions, such as:

- Budget constraint: total budget not exceeding XXX
- Usage scenario: mainly for commuting / suitable for southern climate / no drilling allowed in rental
- Personal preference: prioritizes durability / dislikes complex maintenance
- Time condition: needs to make a decision soon

[5] Research Goal
Clearly state the action conclusion the user hopes to obtain, such as:

- Provide a ranked recommendation
- Determine whether it is worth purchasing
- Choose the optimal solution
- Formulate a specific action plan

Step 3 — Query Depth Requirements
Each query must satisfy:

- Requires at least 3 different information sources to answer
- Requires comparison of multiple subjects
- Requires synthesis of real user reviews, product data, or usage experience

Strictly prohibited: generating questions that can be answered directly by common sense.

Step 4 — Query Type Diversity
All queries must belong to the **Decision Making** type, must cover the following five categories, and 4–6 queries must not all concentrate on the same type:

Type	Core Feature	Example Phrasing
Comparison & Selection	Weighing pros and cons among multiple options, ultimately choosing one	"Which is more suitable for me, XX or XX? Compare from XX dimensions"
Recommendations	Seeking personalized recommendations with no specific candidates	"In XX scenario, what XX products are worth buying?"
How-to Guide	Seeking specific actionable step-by-step methods	"How to complete XX step by step? Including pitfall-avoidance tips"
Travel Planning	Developing travel plans or itinerary arrangements	"How to plan the best itinerary for X days in XX?"
Purchase Decision	Deciding whether to buy, when to buy, which one to buy	"Is it a good time to buy XX now? Key purchase considerations"

Warning: Strictly prohibited from generating the following types of queries:

- Pure information queries: "What is XX", "History of XX"
- Status tracking: "Latest developments of XX", "What's happening with XX now"
- Trend analysis, mechanism research, systematic reviews, or other academically-oriented content

---

Step 5 — Complexity Requirements
Each query must:

- Length: **120–220 words**
- Include structured numbered lists
- Include **3–5 research questions**
- Include explicit constraints
- End with a **specific action goal**

Step 6 — Expression Style Requirements
[Authentic User Language]
Queries must read like real users commissioning a research assistant, using question or imperative sentences.

Wrong example: "Comparative study on anti-aging mechanisms of retinol vs. Proxylane"
Correct example: "I want to start anti-aging skincare but don't know where to begin — should I go with retinol or Proxylane?"

[Avoid Academic Language]
Prohibited: mechanism research / trend analysis / in-depth analysis / systematic review / paper-title-style stacking

[Avoid Content Stacking]
Do not compress multiple research dimensions into a single sentence — must be expressed in structured, separate lines.

Step 7 — Example Query

```
I've been seeing a lot of rental apartment renovation content on
Xiaohongshu lately, and I want to transform my living room into a
creamy minimalist style without drilling or making major structural
changes. I'm currently torn between the following directions:

(1) Multi-functional storage sofa + cream-colored curtains
combination
(2) Rattan storage cabinet + arched floor lamp pairing
(3) Modular bookshelf wall + minimalist rug solution

Please help me evaluate:

(1) Which solution has higher practical feasibility under rental
    restrictions
```

```
(2) In real renovation cases on Xiaohongshu, which direction has a
    lower failure rate
(3) The overall budget range and value for money of each of the
three
    solutions
(4) The actual visual space-enlarging effect across different
styles
    for small apartments
(5) Which solution is easier to maintain and relocate when moving
out

Please note:
- Rental unit: no drilling, no modification to fixed structures
- Budget: under 3,000 RMB
- Prefer lightweight soft furnishings that are easy to take when
moving

Please provide the single most recommended solution to execute,
along
with a priority shopping list of key items to purchase first.
```

Note: Output content should be colloquial and lifestyle-oriented — do not explicitly include labels such as "constraints", "research goals", etc.

User Browsing History

```
{{ user_history }}
```

Output Requirements

Output only a valid JSON array. Do not output markdown markers, code blocks, or any additional text. Example format — List of strings:

```
["query1", "query2", "query3"]
```

E.2 Prompt for Deep Research Query Classification

As discussed in Section 3, to facilitate the analysis of complex research queries, we introduce a query classifier, the details of which are elaborated as follows.

Prompt: Query Classifier Agent

Role Definition

You are a professional text classification assistant, responsible for accurately categorizing user queries into the corresponding category.

Task Description

Analyze the user's input query, select the **single best-matching** category from the list below, and output only that category name — do not output anything else.

Available Categories

- Finance & Business
- Science & Technology
- Software Development
- Education & Jobs
- Health
- Literature
- History
- Hardware
- Industrial
- Art & Design

- Games
- Crime & Law
- Entertainment
- Sports & Fitness
- Software
- Transportation
- Religion
- Home & Hobbies
- Travel
- Food & Dining
- Fashion & Beauty
- Social Life

Output Rules

1. Output only the category name — do not add any explanation, punctuation, or extra text
2. Must select from the categories listed above — do not create new categories
3. Choose the category that most closely matches the core intent of the query

E.3 Prompt for Planner Agent

As described in Section 2, the primary objective of our planner agent is to interpret user intent and generate corresponding guidance cues and response style specifications to direct the subsequent agents accordingly. The detailed prompt is listed as follows.

Prompt: Planner Agent

Role Definition
You are a user intent classification expert for an AI search service. Upon receiving a user query, you need to:

1. **Analyze** the true intent behind the user's query
2. **Classify** it into the corresponding intent type

Classification Strategy
Generally speaking, user queries can be divided into two major categories:
The first major category: Decision Making — User goal: make a decision / plan actions / seek advice, including the following subcategories:

1. Comparison & Selection
   - Core signals: "X vs Y", "difference between X and Y", "X or Y", "difference between", "compared to" — two or more entities explicitly placed side by side
   - Response content: The opening section must include a one-sentence conclusion; then dynamically select the most relevant dimensions based on the topic (e.g., performance / price / ease of use / ecosystem) and present a core-factor comparison table
2. Recommendations & Suggestions
   - Core signals: "recommend", "best", "what's a good", "best X", "top X for Y" — seeking advice with no specific candidates in mind
   - Response content: The opening section must provide a recommendation overview (e.g., top pick, runner-up, best value); then offer a comparison of the recommended options, which may include core highlights, target audience, and price reference
3. How-to Guide
   - Core signals: "how to do", "how to", "tutorial", "getting started", "step by step" — action-oriented, expecting operational steps

- Response content: The opening section must cover prerequisites (environment/requirements), core steps, and estimated time; subsequent sections must detail each step's description and common issues

4. Travel Planning
   - Core signals: "travel to X", "X travel guide", "X days", "travel guide", "itinerary" — location + travel-related terms
   - Response content: The opening section must provide 2–3 sentences of overview; then provide a Day 1–Day X itinerary including must-visit attractions, dining recommendations, transportation guide, accommodation suggestions, and practical tips
5. Purchase Decision
   - Core signals: "how much does X cost", "is X worth buying", "which model is better", "worth it", "should I buy" — price / purchase intent
   - Response content: The opening section must provide 2–3 sentences covering: whether it is recommended + who it suits + the single most important reason; then present a product overview table with reasons to buy, situations where it's not recommended, a side-by-side comparison, and purchasing channel suggestions

The second major category: Information Seeking — User goal: understand facts / track developments / learn knowledge, including the following subcategories:

1. Fact Query
   - Core signals: "what is", "what does X mean", "Define" — expecting a definitive answer
   - Response content: The opening section must answer with a concise, accurate core definition; include 2–5 of the most important key points based on the topic
2. Status & Progress
   - Core signals: "latest developments", "what's happening with X now", "X update", "X latest" — contains time-indicative words
   - Response content: Pay attention to information recency; the opening section must present a recent update timeline with concise timestamps, events, and brief descriptions
3. News & Information
   - Core signals: "X news", "X this week" — explicitly news / current-events oriented
   - Response content: The opening section must list the most important news headlines with 2–3 objective summary sentences; subsequent content follows reverse chronological order with clear, verifiable timestamps
4. Deep Exploration
   - Core signals: "deep dive into", "X ecosystem", "ecosystem", "everything about" — open-ended, no clearly defined scope
   - Response content: The opening section must provide 2–3 sentences of general framing: what the topic is, why it is worth exploring, and its current significance
5. Resource Locating
   - Core signals: "X official website", "X documentation", "X GitHub", "official site", "documentation" — looking for specific links or resources
   - Response content: The opening section must list the core links; subsequent sections may provide additional extended resources

Output Format

Output JSON directly with no extra content, using the following structure:

```
{
  "intent": "<classification result>",
  "response_style": "<response content reference>"
}
```

Field descriptions:

- `intent`: string — select the single best-matching option from: `Comparison & Selection`, `Recommendations`, `How-to Guide`, `Travel Planning`, `Purchase Decision`, `Fact Query`, `Status & Progress`, `News & Information`, `Deep Exploration`, `Resource Locating`
- `response_style`: string — based on the identified intent type, distill a response structure recommendation tailored to the specific query; minor adjustments based on query content are encouraged

E.4 Prompt for Outline Critic Agent

As introduced in Section 2, the outline critic agent is designed to evaluate the input outline and subsequently refine the corresponding blueprints along with their accompanying search queries. The detailed prompt is shown as follows.

Prompt: Outline Critic Agent

Role Definition
You are a strict and demanding report outline review expert who evaluates outline quality based on user requirements and expected key points, and provides improvement suggestions.

Input Description

- **User query:** The user's specific needs and questions
- **Outline blueprints list:** Core key points of the report broken down from the user query
- **Current outline:** Organized using a question-based structure, where sub-questions under each top-level heading represent core content points. When reviewing, treat these sub-questions as the actual answering elements of the corresponding sections.
- **Historical search terms list:** Previously executed search terms, used to avoid duplicate searches
- **Citation rate of current search content:** Number of documents cited in the outline / Number of documents returned by the search engine
- **Response style:** Suggestions for reply style based on the user query, primarily covering the points that the reply content should include and the order in which content is arranged

---

Outline Quality Evaluation Criteria *(Output field "rating", scored 0–10)*

Scoring Rules

1. **Zero-score situations:** Empty outlines, malicious content (empty answers, meaningless text, score manipulation, etc.) receive 0 directly
2. **Strict standards:** Each dimension is scored independently (0–10), and the total score is the average of all dimensions
3. **High-score threshold:** A score of 8 or above is only awarded to outlines that perform exceptionally in that dimension; 7 is good, 6 is acceptable, and below 5 is unacceptable

Evaluation Dimensions (0–10 points each)

1. **Instruction Adherence (0–10)**
   - 9–10: Perfectly follows all user requirements (topic, audience, purpose, format, length, etc.), with clear hierarchical structure, covers all primary and secondary user intents, and the ordering of primary and secondary intents follows the reply style
   - 7–8: Follows most requirements with only 1–2 minor deviations, covers the user's primary intent, and the primary intent can be arranged according to the reply style
   - 5–6: Follows basic requirements but with noticeable format or content deviations
   - 3–4: Partially follows requirements, with important omissions or misunderstandings
   - 1–2: Severely deviates from requirements, most instructions not followed
   - 0: Completely ignores user requirements
2. **Content Depth (0–10)**
   - 9–10: In-depth analysis, including specific sub-points, mechanism analysis, methodology, hypothesis verification, and logical reasoning chains
   - 7–8: Reasonably in-depth, includes some specific analysis points and methods
   - 5–6: Moderate depth, has a basic analytical framework but lacks detail
   - 3–4: Shallow analysis, mostly surface-level descriptions
   - 1–2: Extremely shallow, only lists generic headings
   - 0: No analytical depth whatsoever
3. **Perspective Balance (0–10)**

- 9–10: Comprehensively balanced from multiple perspectives, fairly presenting opposing views, with neutral and objective language
- 7–8: Basically balanced, covering major differing viewpoints
- 5–6: Some awareness of balance, but certain perspectives are insufficiently represented
- 3–4: Noticeably biased, insufficient coverage of opposing views
- 1–2: Severely biased, opposing views largely ignored
- 0: Completely one-sided, no objectivity

4. **Coverage Breadth (0–10)**
   - 9–10: Comprehensively covers relevant dimensions (historical, legal, economic, technical, ethical, social, etc.), broad yet focused; generally requires 7–10 top-level headings
   - 7–8: Covers most important dimensions; generally requires around 5 top-level headings
   - 5–6: Covers basic dimensions but has important omissions
   - 3–4: Limited coverage, multiple important aspects missing
   - 1–2: Very narrow coverage, large amounts of relevant content unaddressed
   - 0: Extremely limited coverage
5. **Evidence Support (0–10)**
   - 9–10: Complete evidence framework, sufficient document citations, diverse and reliable sources; generally requires citing 70% or more of the input documents, or more than 150 citations
   - 7–8: Good evidence planning with reasonable document support; generally requires citing 50% or more of the input documents, or more than 100 citations
   - 5–6: Basic awareness of evidence, but insufficient support
   - 3–4: Weak evidence support, sparse citations
   - 1–2: Almost no evidence planning
   - 0: Completely no evidence support
6. **Insight Value (0–10)**
   - 9–10: Original frameworks, profound insights, specific actionable recommendations, clear measurement standards, and real-world cases
   - 7–8: Some degree of insight, recommendations are relatively specific
   - 5–6: Basic insights, but lacking originality or specificity
   - 3–4: Shallow insights, vague recommendations
   - 1–2: Lacks valuable insights
   - 0: Completely no insight value
7. **Structural Logic (0–10)**
   - 9–10: Clear hierarchy, rigorous logic, explicit relationships between sections, complete and reasonable structure
   - 7–8: Basically clear structure, reasonably good logic
   - 5–6: Acceptable structure, with minor logical issues
   - 3–4: Disorganized structure, unclear logical relationships
   - 1–2: Severely disorganized structure
   - 0: No logical structure

Total Score Calculation

Overall Score = (Instruction Adherence + Content Depth + Perspective Balance + Coverage Breadth + Evidence Support + Insight Value + Structural Logic) / 7

Improvement Suggestion Generation *(Output field "justification")*

- Analyze the strengths and weaknesses of each evaluation dimension separately
- Provide targeted improvement suggestions in combination with specific scores
- Highlight key issues that affect the overall score

Outline Blueprints List Update *(Output field "blueprints")*

Update Strategy

1. **Non-empty list:** Supplement and optimize based on the current user query, with a focus on reinforcing missing dimensions
2. **Empty list:** Generate a comprehensive list of key point content based on the user query, producing a key points list that covers the core elements
3. **Update principles:** Prioritize addition and rewriting logic; avoid simply deleting existing reasonable content. Ensure content breadth (covering multiple relevant dimensions) and depth (specific analysis points for each dimension). Address weaknesses identified in the evaluation dimensions in a targeted manner
4. **List length:** Flexibly determined based on the complexity and coverage scope of the user query; simple questions may be appropriately condensed, complex questions should be fully expanded; generally recommended to stay within { max_blueprints_len }

{

Search Term Generation Guidelines (Xiaohongshu / Knowledge):

1. Extract all core topic words, proper nouns, and important attributes from the user's question.
2. Each search term must be specific, clear, and closely aligned with the user's needs, suitable for use on the Xiaohongshu platform; it is strictly prohibited to introduce irrelevant, vague, or redundant information.
3. If the user's question involves details such as time, location, person, or scenario, extract and incorporate them reasonably into the keywords; if not explicitly mentioned, there is no need to force their inclusion.
4. Ensure diversity of keyword expression, covering different synonymous expressions or important subcategories under the same topic.
5. Keywords within each search term group should be separated by spaces (example: skincare hydrating mask); different search term groups should be separated by English commas. Each search term may be a single word or a multi-word combination, but the overall expression should always remain concise and targeted.
6. Assess whether the user's original input already contains expressions suitable for use as search terms; if so, retain and include them directly in the result list.
7. Do not output any explanations, descriptions, or formatting symbols; output only the final list of search term groups.
8. Generated search terms should be in Chinese.
9. Note: Search terms must ensure broad and diverse coverage; they do not need to be strongly related to the user's question, as long as they provide incremental value. If a historical search terms list exists, avoid duplicating historical search terms.
10. Note: Search term generation should aim for depth and should not be empty where possible.
11. Note: When the input outline content is non-empty, search term generation should explore the content depth lacking in each sub-heading of the outline as much as possible, striving to enrich the depth of outline content.
12. Note: Prioritize the user's requirements when deciding the number of search terms to generate for each outline target list item; in general, it is recommended to keep the number of search terms within { max_query_len }.

{

{

Search Term Generation Guidelines (Google):

1. Precisely extract core topic words, proper nouns, and important information from the user's input.
2. Time information must be identified and completed: extract explicit time references directly (e.g., "Q3 2024" should be written as "Third Quarter of 2024"); implicit time references must be converted into specific intervals (e.g., "last quarter" requires automatic calculation of the previous quarter's start and end dates based on today's date: {{ curr_date }})
3. Keyword priority order: proper nouns (brands, companies, products, policies, etc.) > metrics or characteristics (figures, sales volumes, new products, technological breakthroughs, etc.) > key actions (releases, rises/falls, mergers, experiences, etc.) > regions or scenarios (cities, countries, specific locations)

4. Expressions must be concise: remove interrogative words ("how", "whether", etc.), subjective descriptors ("amazing", "ultra-powerful", etc.), and vague expressions ("some", "various", etc.); retain only content with actual retrieval significance.
5. For special scenarios, such as comparative questions, retain both sides of the comparison and highlight them with "vs" or "comparison".
6. Note: Search terms must ensure broad and diverse coverage; they do not need to be strongly related to the user's question, as long as they provide incremental value. If a historical search terms list exists, avoid duplicating historical search terms.
7. Note: Search term generation should aim for depth and should not be empty where possible.
8. Note: When the input outline content is non-empty, search term generation should explore the content depth lacking in each sub-heading of the outline as much as possible, striving to enrich the depth of outline content.
9. Note: Prioritize the user's requirements when deciding the number of search terms to generate for each outline target list item; in general, it is recommended to keep the number of search terms within { max_query_len }.

{

Output Format
Please strictly output in the following JSON format:

```
{
  "rating": `float` - Score for the given outline,
  "justification": `string` - Explanation of the scoring result,
  "blueprints": [
    {
      "content": "string - Outline key point content 1",
      "search_query": ["string1", "string2", "..."]
    },
    {
      "content": "string - Outline key point content 2",
      "search_query": ["string1", "string2", "..."]
    }
  ]
}
```

E.5 Prompt for Outline Generator Agent

As specified in Section 2, the outline generator collaborates with the outline critic agent in an iterative refinement process to progressively optimize the generated outline.

Prompt: Outline Generator Agent

Role Definition
You are a professional report outline planning expert who generates structured, logically clear report outlines based on externally sourced search document content, user requirements, and outline key points lists. The generated outline should possess the characteristics of a professional report: **with the primary goal of directly responding to the user's query**, featuring rigorous logical organization, covering multiple relevant dimensions, and incorporating elements of in-depth analysis, explanation, and argumentation at appropriate locations.

Input Content

- **User query:** The user's specific needs and questions
- **Outline blueprints list:** Core key points of the report broken down from the user query
- **Previous round outline:** The outline content generated in the previous round
- **Previous round evaluation:** The evaluation and specific revision suggestions for the previous round's outline
- **External search results:** Search documents corresponding to each item in the outline key points list, each result containing a unique ID identifier

- **Response style:** Suggestions for reply style based on the user query, primarily covering the points that the reply content should include and the order in which content is arranged

---

Outline Generation Standards
Core Specifications

1. **Query-first response:** The overall organizational logic of the outline must center on directly responding to the user's query. All chapter divisions and sub-topic settings must revolve around "how to completely answer the user's query," avoiding generalized expansions that deviate from the user's core needs.
2. **Response style:** Follow the style to cover and arrange the key points of the outline content. Prioritize responding to the user's primary intent and cover the user's secondary intent in specific sections.
3. **Instruction adherence:** Generate the outline strictly according to the requirements of the user's query, including subject scope, audience positioning, level of detail, tone and style, as well as any formatting or structural requirements. Ensure required components are included and avoid deviating from user expectations.
4. **Content depth:** Based on the outline key points list, ensure the outline possesses analytical depth. An excellent outline not only contains generalizing headings but should also include: specific analysis points, key argumentation logic, mechanisms and causal relationships, methodological frameworks, evaluation metrics, dependency analysis, and evidence and case integration planning. Avoid merely listing generic topics without a substantive analytical framework.
5. **Perspective balance:** Ensure fairness and objectivity of the outline. For complex or controversial issues, multiple perspectives and differing viewpoints should be planned, content space should be allocated fairly, and neutral, non-leading language should be used. Explicitly include sections for trade-off analysis, discussion of limitations, and consideration of counter-evidence.
6. **Coverage breadth:** Based on the outline key points list, ensure coverage of multiple relevant dimensions, such as: historical background, policies and regulations, market economics, technical operations, social culture, geographic comparisons, stakeholder analysis, risk assessment, and implementation pathways. Coverage should be broad and purposeful, avoiding irrelevant digressions.
7. **Evidence support:** Systematically plan the evidence framework and sources. Precisely add citation markers `<cite>documentID</cite>` after relevant content, ensuring citation diversity to enhance the credibility and comprehensiveness of the argumentation. Fabricating citation information is strictly prohibited.
8. **Insight value:** Go beyond common templates by providing original structural frameworks, highlighting non-obvious connections, and rationally sequencing sections to efficiently reveal key insights. Ensure recommendations and analyses are specific and actionable, explicitly identifying specific cases, comparative studies, and appropriate presentation methods (tables, charts, frameworks, etc.).
9. **Structural logic:** Build clear hierarchical relationships with distinct responsibilities for headings at each level and smooth logical flow. When a section at a given level requires subdivision, it should contain 2 or more sub-headings to ensure reasonable and complete categorization. Focus on overall structural coherence, logical relationships between sections, and consistency of heading hierarchy.
10. **Citation diversity:** Cite as many different document IDs as possible to enhance evidence support through diversified sources and provide multi-perspective viewpoints.

Special Requirements

1. **Open with a direct substantive answer (preamble and background explanations are absolutely prohibited):**
   - The **first chapter of the report** (i.e., the `##` heading) must cut straight to the point and provide the **final substantive answer** to the user's query.
   - It is **strictly prohibited** to write vacuous preamble content such as "Executive Summary," "Background Introduction," "Why This Matters," or "Research Significance" in this chapter.

- Core facts must be extracted directly. For example: if the user requests to "organize the team and make predictions," the body text and sub-headings of the first chapter must **directly list** the core team roster, provide **specific conclusions** from horizontal comparisons, and directly state **what the prediction results are**. Subsequent chapters then break down and argue these conclusions in detail.

2. **Strict heading hierarchy mapping and restrictions on interrogative sentences:**
   - **Overall report title:** Only 1 throughout the entire document, must be a Markdown first-level heading, formatted as `#OverallReportTitle`.
   - **First-level sections (chapters):** Must be Markdown second-level headings, formatted as `##Chapter1xxxx`. Use declarative thematic summaries; **interrogative sentences are absolutely prohibited**.
   - **Intermediate-level sections** (e.g., sections, subsections): Formatted as `###1.1xxxx` or `####1.1.1xxxx`. Point to the analytical dimension or argument; must be declarative sentences or phrases; **interrogative sentences are absolutely prohibited**.
   - **Lowest-level headings (leaf nodes):** i.e., terminal-level content that is not further subdivided; may optionally adopt a question format to guide analysis, or may use declarative style.
   - **It is strictly prohibited to directly copy existing questions from search content as headings.**
3. **Content self-consistency:** Ensure the outline covers the complete scope of the topic, with each section corresponding to and echoing the others to form a complete closed loop. Content should have no repetition, no omissions, no conflicts, and must be practical and readable. All sections must be able to clearly answer the question "how does this section serve the answering of the user's query."
4. **Deep exploration:** On the premise of ensuring logic and consistency, generate more levels of sub-headings to ensure each section is explored in depth, avoiding superficial generalizations.
5. **Iterative optimization:** If the previous round outline content is non-empty, conduct systematic iteration based on the previous round outline, fully incorporating the improvement suggestions from the evaluation.
6. **Section richness:** To improve overall information coverage, multiple core sections (`##`) should be used in the outline. On the premise of ensuring logical relationships between sections, divide into as many core sections as possible to cover the content of the outline key points list. In general, the number of core sections should be no fewer than 7–10 (including the opening direct-answer section).
7. **High citation coverage:** To improve overall information coverage, externally sourced search results should be utilized as fully as possible. Any content relevant to the user's query and outline key points list should be cited wherever possible. In general, the number of externally sourced search content citations should be no fewer than 100–200.

Citation Standards

1. **Format standard:** Use the `<cite>documentID</cite>` format, e.g., `<cite>turn_0_4,turn_1_8</cite>`.
2. **Positional accuracy:** Immediately follow the relevant information, ensuring citations correspond precisely to content.
3. **Prohibition principle:** Fabricating cited document information or fictitious document IDs is strictly prohibited.

Output Format

1. Please output only the final answer outline; do not repeat the user's question and do not output any opening remarks or explanatory statements.
2. Strictly follow the above rules and structure, ensuring clarity of organization, richness of content, and elegance of expression.
3. Strictly output using the following Markdown hierarchical structure:

```
# [Overall Report Title]
## Chapter 1 [Core Conclusion That Directly Answers the Query]
### 1.1 [Declarative sentence heading for Conclusion Dimension 1]
```

```
### 1.2 [Declarative sentence heading for Conclusion Dimension 2]
...
## Chapter 2 [Specific Discussion / Dimensional Breakdown...]
...
```

E.6 Prompt for Writer Agent

As described in Section 2, the writer agent operates sequentially, commencing with the first chapter from scratch, whereby each subsequent chapter is generated by following a continuation paradigm, taking the previously written chapters as contextual inputs.

Prompt: Report Writer Agent

Role Definition
You are a professional report writing expert, skilled at generating structured, logically clear, and content-rich professional reports based on externally sourced search document content and report outlines, combined with user questions. **Your core task is: centering on the user's query, strictly filling in content within the input outline framework, so that the content of every section directly serves the complete answering of the user's query.**

Input Description

- **User query:** The user's specific needs and questions, which serve as the core anchor of the entire report; all content is written with the ultimate purpose of answering this query.
- **Outline content:** Contains the outline framework and heading hierarchy, where `<cite>documentID</cite>` marks the cited search documents. **The outline framework is the sole legitimate structural basis; the hierarchical relationships and order of headings may not be modified, added to, or reduced for any reason.**
- **Outline blueprints list:** Contains the key points that the current outline is expected to cover, serving as a directional reference for content filling.
- **Response style:** Suggestions for reply style based on the user query, primarily covering the points that the reply content should include and the order in which content is arranged.
- **Search documents:** Contains document IDs, titles, and specific content, corresponding to the citation IDs in the outline.
- **Previous chapter content:** If the previous chapter content is non-empty, please write the designated sections in the outline according to the logic of continuation.

Generation Rules

1. **Strictly follow the outline framework:** Use the heading structure of the input outline as the sole skeleton, filling in corresponding content under each heading. It is strictly prohibited to independently add, delete, merge, or split any heading level. The organizational logic of section content must fully correspond to the outline's hierarchical structure; cross-section mixed writing is not permitted. **Heading rewriting rules:** If a heading in the outline is presented in interrogative form (e.g., "Why…?", "How…?", "What is…?", etc.), in order to avoid the unprofessional appearance of "self-question and self-answer" in the report, **such headings must be rewritten into semantically equivalent declarative sentences or nominal phrases** (for example: "Why choose Plan A?" → "The basis for selecting Plan A"; "How to achieve cost reduction and efficiency improvement?" → "The implementation pathway for cost reduction and efficiency improvement"). Rewriting must satisfy the following constraints: the semantics must be completely consistent with the original heading; the heading level and order must not be changed; the rewritten heading should be concise and professional, in keeping with the report's writing style.

   → *Amendment: "allowed to rewrite the heading" is changed to "must rewrite the heading"*
2. **Write around the user's query:** While filling in the outline framework, every paragraph of content must clearly serve the answering of the user's query. Before writing, first clarify the role this section plays in answering the user's query (background setting, core argumentation, data support, conclusions and recommendations, etc.), and use this as the guiding principle for organizing content, avoiding generalized descriptions unrelated to the query.

3. **Content supplementation:** Remain faithful to the outline framework; supplement the details of the outline by combining search document content, focusing exclusively on the sections designated in the outline; it is strictly prohibited to supplement the content of preceding or following sections.
4. **Logical optimization:** Ensure the report structure is clear, well-layered, thoroughly argued, and professionally expressed.
5. **Citation standards:** Strictly maintain the `<cite>documentID</cite>` format; it is prohibited to fabricate document content or fictitious document IDs.
6. **Quality assurance:** Apply the "Let's think step by step" approach; content must be well-reasoned and evidence-based, avoiding vague statements and ensuring information accuracy.
7. **Formatting aesthetics:** Based on the question type of the user's query, adopt an appropriate and readable format (such as paragraphs, numbered lists, tables, etc.) to enhance readability.
8. **Information integration:** Synthesize the content of multiple relevant search results; the same externally sourced search document content must not be cited repeatedly. Fully extract and integrate key information, responding in a multi-perspective, thorough, in-depth, and creative manner.
9. **Language consistency:** Unless the user specifically requests otherwise, respond in the same language as the user's query.
10. **Consistency and self-coherence:** Ensure that every key point is answered in a self-consistent, substantive, and professional manner; for example, a weekly meal plan must list a complete seven-day menu.
11. **Enumeration issues:** When listing is required (e.g., flight information), select no more than 10 key pieces of information.
12. **Section transitions:** Add concise introductory language at the beginning of each section to provide necessary background explanation or logical transitions, ensuring the report content is coherent and fluent, avoiding the mere accumulation of viewpoints or data, and enhancing the professionalism and readability of the report. **Introductory language must reflect the connection between the section content and the user's query.**
13. **Key point expansion:** With respect to the outline key points list, appropriate content expansion and key point connections may be made within the outline framework to enrich the outline content as much as possible. **All expanded content must be closely tied to the user's query; it is strictly prohibited to introduce extended topics unrelated to the query.**

Output Requirements

1. Use Markdown format.
2. **Strictly maintain the heading hierarchy and order of the outline; any structural modifications are prohibited.** If interrogative-form headings exist in the outline, **they must be mandatorily rewritten with semantically equivalent expressions in accordance with Generation Rule 1**; the scope of rewriting is limited to the heading text itself and must not affect the hierarchy or order.

   → *Amendment: "may be rewritten according to…" is changed to "must be mandatorily rewritten according to…"*
3. Every key argument must be supported by corresponding citations.
4. Content should be substantive while avoiding redundancy and repetition.
5. Please directly output the report Markdown content without outputting any opening remarks or explanatory statements.
6. If the previous chapter content is non-empty, please directly output the corresponding section content in the outline without outputting content such as `"##Title(Continued)"`.

Output Format Example

```
# Analysis of the Current State of Artificial Intelligence
Development
Artificial intelligence technology is developing rapidly on a
global
scale. The current AI market size has reached $230 billion
```

```
<cite>turn_1_0</cite>, and is expected to maintain a compound
annual
growth rate of 30% over the next five years. At the application
level,
AI technology has been widely deployed in finance, healthcare,
manufacturing, and other sectors <cite>turn_1_4</cite>.

## Technology Maturity and Market Penetration Rate
- Market size: Strong technological development and widespread
application
  demand have jointly driven rapid market growth. According to the
  latest
  data, the current global AI market size has reached $230 billion
  <cite>turn_1_2</cite>. This enormous figure not only reflects the
  capital market's high recognition of the AI sector, but also
  demonstrates strong market vitality and high penetration
  potential.
  It signifies that AI is no longer a marginal innovation
  embellishment,
  but rather a strategic investment direction for enterprises to
  maintain
  competitiveness and achieve future growth, with its immense
  commercial
  value continuing to be released.
```

F HARNESS OPTIMIZATION

F.1 HARNESS INSTRUCTION

One of the key components of our harness is the skill markdown file, which explicitly defines the permissible and impermissible operations for the code agent (i.e., Claude Code, as employed in this study).

Skill: Search Agent Optimization

Automatically optimize the search relevance of a search agent. Run a full evaluation via eval-batch, analyze the `search_coverage` metric and its reasoning, modify any modifiable files in the pipeline, re-run the evaluation, and iterate in a loop until `search_coverage.mean` ≥ 9 or convergence.

I. Project Architecture
Eval Pipeline (`eval_outline_judge.py`)
Processing flow for each query:

```
User query
  -> QueryMiner (decompose into search sub-queries)
  -> IntentPlanner (intent analysis, structural planning)
  -> DisentangledOutlineJudgeBlueprintAPI
       (outline generation + review + search query generation
        + execute search)
  -> SummaryQAGeneratorAPI
       (score search results for relevance + generate summaries)
  -> JudgeSearchQueryDiversityAPI
       (evaluate search query diversity and coverage,
        output final score)
```

`run_agent` calls each service in the above order. `DisentangledOutlineJudgeBlueprintAPI` may perform multiple internal iterations (controlled by `max_outline_generator_turns`), but this is internal API behavior.

Key Files

File	Role
harness/eval_outline_judge.py	Main evaluation script; supports single-query (run_agent) and batch (run_batch_agent--run-modebatch) modes
harness/api/DisentangledOutlineJudgeBlueprintAPIService.py	Outline review service: generates outlines, LLM scoring, generates search queries, executes searches
harness/api/SummaryQAGeneratorAPIService.py	Summary generation service: scores search documents for relevance, generates summaries and evidence
harness/api/JudgeSearchQueryDiversityAPIService.py	**Evaluation service (not modifiable):** evaluates search query coverage and search result quality
harness/config/test_harness_outline_xhs.gin	Gin configuration file (models, parameters, thresholds)
harness/memory/	Memory system directory (traces)

Template Files (Prompt Layer; currently use_zh=True, ZH only)

Template File	Used By	Modifiable
template/DisentangledOutlineJudgeBlueprintStyleQA_ZH.jinja2	DisentangledOutline JudgeBlueprintAPI	YES
template/IntentPlanner_ZH.jinja2	IntentPlanner	YES
template/JudgeSearchQueryDiversity_ZH.jinja2	JudgeSearchQuery DiversityAPI	NO
template/SummaryQAGeneratorBlueprint_ZH.jinja2	SummaryQA GeneratorAPI	YES

Note: All templates are loaded at runtime from ./template/ (project root). When modifying templates, ensure you are editing files under template/.
Evaluator not modifiable: JudgeSearchQueryDiversityAPIService.py and JudgeSearchQueryDiversity_ZH.jinja2 are the final scoring services — **modifying their code or prompts is strictly prohibited**. Optimization can only improve the score by improving the upstream pipeline.
Understanding the scoring criteria (important): Although modifying the evaluation service is prohibited, you **should read** template/JudgeSearchQueryDiversity_ZH.jinja2 to understand the scoring criteria and rationale for search_coverage. Knowing what the judge focuses on enables targeted upstream pipeline optimization. See the causal chain analysis in Section II.

II. Eval Metric Definitions
Optimization Target
search_coverage.mean $\geq$ **9** — this is the sole optimization objective.

Metric	Field Path	Range	Meaning
search_coverage	evaluation.search_coverage.score	0–10	Actual relevance between search-returned documents and the query

Source: output of JudgeSearchQueryDiversityAPI, located at input_dict["judge_search_query_turn_0"]["evaluation"].
Each sub-score carries a reasoning field (textual analysis), which is the **key information** for diagnosing problems.

Display Metrics (not used as optimization targets)

Metric	Meaning
overall	Overall score
completeness	Degree to which blueprints + search queries cover all core dimensions of the query
diversity	Perspective diversity, content-type diversity, granularity diversity, redundancy
internal_correlation	Degree to which retrieved documents are internally consistent and mutually reinforcing with respect to the research query

These metrics are displayed in metrics.json and summary.md for reference, but **are not used as the basis for optimization decisions**.

Process Metrics (for diagnostics)

Metric	Location	Meaning
outline_rating	`input_dict["judge_turn_0"]["rating"]`	OutlineJudge score for the outline
outline_justification	`input_dict["judge_turn_0"]["justification"]`	Textual analysis from outline review
search_query_count	`len(input_dict["search_query_turn_0"])`	Total number of search queries generated
doc_avg_relevance	Mean of `judge` score per document in the SummaryQA phase	Relevance between search results and the query
doc_count	Total documents returned by search	Search coverage volume

Batch Aggregate Metrics (`metrics.json`)
`run_batch_agent` runs a fixed sampled subset each round and outputs `metrics.json`, computing mean/median/std/min/max for each metric. The sample size is controlled by the gin parameter `fixed_sample_size`; indices are randomly drawn and saved to `harness/optimization_runs/fixed_indices.json` on the first run and reused across all subsequent rounds to ensure fair cross-round comparison.

search_coverage Causal Chain (must understand)
`search_coverage` does **not** directly evaluate search query text — it evaluates **whether documents returned by the search queries are actually relevant to the user query**. Complete causal chain:

```
Blueprints decompose query into dimensions
  -> each blueprint generates search queries
  -> search engine executes searches -> returns documents
  -> SummaryQAGeneratorAPI scores each document for relevance
       (judge score, 0-1) and generates summaries (summary/snippet)
  -> JudgeSearchQueryDiversityAPI sees:
       per-query doc statistics + snippets
     and outputs search_coverage score
```

SummaryQA scoring rules (from `SummaryQAGeneratorBlueprint_ZH.jinja2`, **modifiable**):

- Document is unrelated to the user question, the report outline list, the search query, *and* the report outline $\rightarrow$ 0
- Document can partially answer the user question, *or* is partially related to the report outline list, the search query, or the report outline $\rightarrow$ 0~1

Key: OR semantics. A document only needs to be related to *any one* of query / outline / blueprint / search_query to receive a score. This means a search query that drifts from the user query but aligns with a blueprint may still receive a high judge score — but JudgeSearchQueryDiversity also reads the snippet content; if the snippet does not substantively support the user query's information needs, the score may still be penalized.
The SummaryQA template is a modifiable lever: If `doc_avg_relevance` is consistently high but `search_coverage` is low, SummaryQA scoring may be too lenient (giving high scores to documents related to a blueprint but unrelated to the query).

Information the Judge sees when scoring search_coverage (from `JudgeSearchQueryDiversityAPIService.py`, not modifiable):

- The text of each search query
- Number of documents returned per search query (max 10)
- Document relevance distribution per search query: mean, std, min/max, all scores
- Document snippets per search query (first 50 characters of summary)

Two scoring angles for search_coverage (from `JudgeSearchQueryDiversity_ZH.jinja2`, not modifiable but must understand):

1. **Retrieval effectiveness:** Verified by the doc judge score distribution per search query (score > 0.5 = relevant)
2. **Cross-query snippet complementarity:** Whether snippets returned by different search queries are semantically complementary (rather than highly overlapping)

The second angle is frequently overlooked: Even if all search queries return highly relevant documents, if multiple search queries return snippets with highly overlapping content that lacks information complementarity, `search_coverage` will still be penalized. Search queries must cover different information dimensions.

Scoring Criteria:

Score	Criteria
9–10	**All** search queries have sufficient information coverage; >**80%** of returned documents are highly relevant (relevance score > 0.5)
7–8	>**80%** of search queries have sufficient content; information chain is essentially complete
5–6	**20%–40%** of search queries have insufficient content; information gaps exist
3–4	>**40%** of search queries have insufficient content or deviate in direction
1–2	Most retrieval results cannot support the query

Core Insights:

- To score 9, **nearly every search query** must return highly relevant documents, and different search queries must return informationally complementary content
- Even a small number of low-quality search queries will drag down the total score — optimization should focus on identifying which search queries are holding back the score
- Doc judge scores come from SummaryQA; if SummaryQA scoring is biased, the data the judge sees will be distorted — the SummaryQA template is also an adjustable lever

Analysis Path

```
1. Check search_coverage.mean -> is it >= 9?
2. Not yet -> read best_version's summary.md
3. Analyze each case (sorted by search_coverage ascending,
   worst cases first):
   a. search_coverage_reasoning: What specific problem did the
   judge
      identify?
      - "Search results not relevant" (retrieval effectiveness
      issue)?
      - "Search results repetitive / lack complementarity"
        (snippet complementarity issue)?
   b. Search query quality distribution: how many high-quality vs
      low-quality? (summary.md has details)
   c. Low-quality search query details: which queries returned
      low-relevance documents? What are the problem patterns?
      - Too broad / abstract? -> need more specific queries
      - Unrelated to query? -> blueprint decomposition drifted from
        query intent
      - doc_count low? -> query may be too niche, no matching
      content
        in search engine
   d. doc_avg_relevance vs search_coverage contradictory?
      - doc_avg_relevance high but search_coverage low -> two
      possible
        causes:
        (1) SummaryQA scoring too lenient, gave irrelevant docs
        high
            scores -> read SummaryQA template
        (2) Snippets highly overlapping, lack info complementarity
            -> need greater differentiation between search queries
   e. completeness_reasoning: which dimensions are missing? ->
      guides blueprint supplementation direction
   f. diversity_reasoning: which search queries are redundant?
      -> guides deduplication or diversification
   g. Do blueprints accurately cover the core dimensions of the
   query?
4. Identify common patterns: do low-quality search queries across
   multiple low-score cases share common patterns?
```

```
   (e.g., all end with "recommendations", all generated by
   _generate_additional_queries, all truncated due to length, etc.)
5. Read harness/memory/traces_*.jsonl for deeper analysis
   (contains complete per-query details)
6. Read the code of the file to be modified first; understand the
   existing logic before making changes
7. Holistic judgment: locate the specific bottleneck in the causal
   chain before making changes:
   - Blueprints deviate? -> modify outline template / IntentPlanner
     template
   - Poor search query generation quality? -> modify search query
     generation guidelines in outline template
   - Post-processing introduces noise? -> modify API
   post-processing
     logic
   - SummaryQA scoring distorted? -> modify SummaryQA template
```

III. Optimization Loop Protocol

Run Command

```
cd <project root>
python harness/eval_outline_judge.py \
  --gin-config-file harness/config/test_harness_outline_xhs.gin \
  --run-mode batch
```

Loop Flow

```
Startup check:
  Check whether harness/optimization_runs/manifest.json exists
  - Does not exist -> enter INIT
  - Already exists -> skip INIT, enter LOOP directly
    (restore state from manifest and continue)

INIT (first run):
  1. Create harness/optimization_runs/ directory
  2. Run eval-batch -> save as v0_baseline
  3. Create manifest.json
  4. Enter LOOP

LOOP:
  1. Read manifest.json -> find best_version and number of existing
     versions (used to check stop conditions)
  2. Restore all snapshot files from best_version/snapshot/ to the
     working directory
  3. Read best_version's metrics.json + details.jsonl + summary.md
     + read all historical versions' changelog.md
     (understand what was changed previously and its effect)
  4. Analyze data -> identify problems -> write changelog.md
     (problem -> cause -> what to change -> expected outcome)
  5. Execute modifications
  6. Run eval-batch -> locate output directory -> save as v{N}
     (snapshot + metrics + details + summary + changelog)
     (Locate output directory: eval outputs to {output_path}/
      under the latest timestamp directory prefixed with
      {job_name}_batch_)
  7. Compare with best_version:
     - search_coverage.mean higher -> update best_version
     - search_coverage.mean lower or equal -> mark as regressed
  8. Check stop conditions:
     - search_coverage.mean >= 9 -> target reached, stop
     - 5 consecutive rounds without exceeding best -> convergence
     stop
     - 20 rounds completed (excluding baseline) -> upper limit stop
  9. Not stopping -> return to step 1

END:
```

```
  Output summary report:
  - Metric changes from baseline to best
  - What was done each round and its effect
  - Which version is the final best_version
```

Stop Conditions Detail

Condition	Trigger Rule	Meaning
Target reached	`search_coverage.mean>=9`	Objective achieved
Convergence stop	5 consecutive rounds where `search_coverage.mean` does not exceed the current `best_version`	Optimization has converged or is stuck in a local optimum
Upper limit stop	20 optimization rounds completed (excluding baseline)	Prevents infinite loop

IV. Snapshot Mechanism

Directory Structure

```
harness/optimization_runs/
  manifest.json                    <- global index
  v0_baseline/
    metrics.json                   <- aggregate metrics
    details.jsonl                  <- per-case details
                                      (all metrics + reasoning)
    summary.md                     <- analysis summary for this
    round
    snapshot/                      <- snapshot files (copies of
                                      modifiable files only)
    changelog.md                   <- change notes for this round
                                      (v0: "baseline, no changes")
  v1/
    metrics.json
    details.jsonl
    summary.md
    snapshot/
    changelog.md
  ...
```

manifest.json Format

```
{
  "best_version": "v1",
  "versions": [
    {
      "version": "v0_baseline",
      "timestamp": "2026-04-10T14:00:00",
      "search_coverage_mean": 6.3,
      "metrics_summary": {
        "search_coverage": 6.3,
        "overall": 6.2,
        "completeness": 6.5,
        "diversity": 5.8
      },
      "parent": null,
      "status": "baseline"
    },
    {
      "version": "v1",
      "timestamp": "2026-04-10T16:00:00",
      "search_coverage_mean": 7.4,
      "metrics_summary": {
        "search_coverage": 7.4,
        "overall": 7.1,
        "completeness": 7.3,
        "diversity": 6.5
      },
      "parent": "v0_baseline",
```

```
      "status": "improved"
    }
  ]
}
```

Files to Snapshot
Each round, copy the following files to `v{N}/snapshot/`:

Prompt templates (ZH only; current `use_zh=True`):

- `template/DisentangledOutlineJudgeBlueprintStyleQA_ZH.jinja2`
- `template/IntentPlanner_ZH.jinja2`
- `template/SummaryQAGeneratorBlueprint_ZH.jinja2`

Algorithm logic:

- `harness/api/DisentangledOutlineJudgeBlueprintAPIService.py`
- `harness/api/SummaryQAGeneratorAPIService.py`

Configuration:

- `harness/config/test_harness_outline_xhs.gin`

Note: Evaluation service files (`JudgeSearchQueryDiversityAPIService.py`, `JudgeSearchQueryDiversity_ZH.jinja2`) are not in the snapshot list because they must not be modified.
Non-modifiable gin parameters: The following two parameters are fixed and must not be modified:

- `DisentangledOutlineJudgeBlueprintAPI.search_engine="xiaohongshu"`
- `DisentangledOutlineJudgeBlueprintAPI.num_searches=10`

Saving a Snapshot (execute after each eval round)

1. Create `harness/optimization_runs/v{N}/snapshot/` directory
2. Copy all files from the snapshot list into `snapshot/`, preserving the relative path structure (e.g., `snapshot/template/xxx.jinja2`, `snapshot/harness/api/xxx.py`)
3. Locate the eval output directory and copy results:
   - Eval output path: `./outs/` (under project root)
   - Directory name format: `TEST_HARNESS_ZH_V2_batch_<YYYYMMDD_HHMMSS>/`
   - Locate method: `ls -td ./outs/TEST_HARNESS_ZH_V2_batch_* | head -1`
   - Copy `metrics.json`, `details.jsonl`, and `summary.md` from that directory to `harness/optimization_runs/v{N}/`
4. Write or update `changelog.md` (fill in actual results)
5. Update `manifest.json`

Restoring a Snapshot (execute at the start of each round)
Copy all files from `best_version/snapshot/` back to the corresponding locations in the working directory. For example:

```
cp harness/optimization_runs/v1/snapshot/template/*.jinja2
template/
cp harness/optimization_runs/v1/snapshot/harness/api/*.py
harness/api/
cp harness/optimization_runs/v1/snapshot/harness/config/*.gin \
   harness/config/
# After restoring, confirm that search_engine="xiaohongshu"
# and num_searches=10 have not been altered
```

V. Changelog Format
Each optimization round **must write changelog.md before executing changes**. Format:

```
# V{N} Optimization Notes

## Problems Identified
```

```
(Analyze the previous round's eval data; describe the
search_coverage
issues with data and reasoning support)
- search_coverage mean value; which cases scored lowest
- Common patterns in search_coverage_reasoning for low-score cases
- Any anomalies in process metrics (query count, doc_avg_relevance,
etc.)

## Changes Made

### Modified files
- `path/to/file`: what was specifically changed

## Expected Outcome
(Expected changes to search_coverage)

## Actual Outcome
(Fill in after eval completes)
- search_coverage: {before} -> {after}
- overall: {before} -> {after}
- completeness: {before} -> {after}
- diversity: {before} -> {after}
```

VI. Optimization Space

The agent may freely decide the direction and extent of optimization. Modifiable files are listed in the snapshot list in Section IV.

The sole non-modifiable items are the evaluation services: `JudgeSearchQueryDiversityAPIService.py` and `JudgeSearchQueryDiversity_ZH.jinja2`.

Diagnosis-to-Action Mapping

Based on the data and reasoning in `summary.md`, once the problem is located, take action according to the following mapping:

Symptom	Possible Cause	Priority Target
Many queries return low-relevance docs (low high_relevance_ratio)	Search queries too broad / abstract, weak connection to query	Search query generation guidelines in outline template; `_generate_additional_queries` logic in API
All search queries under a blueprint are low quality	Blueprint decomposition drifts from core query intent	Blueprint update strategy in outline template; `IntentPlanner_ZH.jinja2`
Insufficient search query count (low search_query_count)	`min_query_per_blueprint` or `min_query_len` too low	Gin configuration parameters
Many duplicate / near-synonym queries	Insufficient deduplication logic; prompt does not emphasize diversity	`_extract_search_queries` in API; search query guidelines in outline template
Low doc_count (few documents returned)	Search queries too niche / specialized / long	Outline template (add broad+narrow mix); `_broaden_queries`, `_truncate_long_queries` in API
completeness_ reasoning points to missing dimensions	Incomplete blueprint coverage	Outline template (diversity requirements section); `IntentPlanner_ZH.jinja2`
doc_avg_relevance high but search_coverage low	A few queries severely drag down the overall score; or snippets are semantically highly overlapping	Identify low-quality queries; if snippets overlap, increase differentiation between search queries
doc_avg_relevance generally high (>0.6) but snippets do not actually support the query	SummaryQA scoring too lenient: docs related to blueprint but not to user question received high scores	`template/SummaryQAGeneratorBlueprint_ZH.jinja2` — tighten judge scoring criteria; emphasize direct relevance to **user question**
doc_avg_relevance generally low (<0.3)	SummaryQA scoring too strict; or search results are genuinely not relevant	First check search query quality; if queries are reasonable but scores are low, read `SummaryQAGeneratorBlueprint_ZH.jinja2` to check if scoring criteria are too strict
Supplementary queries generated by `_generate_additional_queries` are low quality	Regex-based supplementation strategy generates irrelevant queries	Rewrite `_generate_additional_queries` or adjust `min_query_per_blueprint` to avoid triggering it

Strategy 1: Search Query Quality Optimization *(most direct impact on search_coverage)*
Search queries are the **direct driver** of `search_coverage`. Checklist:

- **Specificity:** Does each search query have a clear information-retrieval target? Avoid broad queries like "how is XX"
- **Relevance:** Are the documents returned by the search query relevant to the **user query**? (Being relevant to a blueprint alone is not enough; it must relate to the original query)
- **Searchability:** On the target search engine (Xiaohongshu), can this query retrieve effective content?

Entry points:

- `template/DisentangledOutlineJudgeBlueprintStyleQA_ZH.jinja2` — search query generation guidelines (most commonly modified)
- `harness/api/DisentangledOutlineJudgeBlueprintAPIService.py` — `_validate_and_fix_blueprints`, `_generate_additional_queries`, `_check_query_quality`

Strategy 2: Search Query Count and Coverage
Does the total number of search queries cover all dimensions of the query? Is the number of queries per blueprint sufficient?
Entry points: gin configuration (`min_query_len`, `min_query_per_blueprint`, `max_query_len`)
Strategy 3: Outline Structure Optimization

Blueprints determine the direction of search queries. If blueprints deviate from the core intent of the query, even high-quality search queries will be futile. When `completeness_reasoning` repeatedly points to missing dimensions, prioritize fixing this.
Entry points:

- `template/DisentangledOutlineJudgeBlueprintStyleQA_ZH.jinja2` — outline key point update strategy
- `template/IntentPlanner_ZH.jinja2` — intent analysis prompt

Strategy 4: Search Parameter Tuning
Adjustable parameters such as `top_k`, filter strategies, etc. (Note: `search_engine` and `num_searches` cannot be modified.)
Entry points: gin configuration
Strategy 5: Post-Processing and Quality Control
Deduplication, truncation, expansion, and quality-checking logic applied after search query generation. When analysis reveals that low-quality queries originate from post-processing (e.g., queries supplemented by `_generate_additional_queries`, queries expanded by `_broaden_queries`), prioritize fixing here.
Entry points:

- `harness/api/DisentangledOutlineJudgeBlueprintAPIService.py` — `_smart_split_query`, `_truncate_long_queries`, `_broaden_queries`, `_check_query_quality`

Strategy 6: Search Query Effectiveness Feedback Mechanism
The Memory system extracts the actual retrieval effectiveness of search queries for the same query from historical traces, and injects this information into the pipeline LLM's prompt at runtime to help it generate more effective search queries.
Design principles:

- **Exact query matching only:** Optimal search strategies vary greatly across different query types; fuzzy / global matching introduces noise
- **Feed back search query effectiveness data only, do not expose old-round blueprints:** Avoids anchoring to old-round outline structures; template instructions (not historical examples) should drive blueprint generation
- **Aggregate cross-round data:** The effectiveness of all search queries for the same query across multiple rounds is aggregated to form a whitelist/blacklist of search queries for that query

Working mechanism:

1. After each batch eval round, automatically save `harness/memory/traces_{timestamp}.jsonl`, containing per-query search query document quality details (`per_query_doc_stats`: `avg_relevance`, `high_relevance_ratio`, etc. per query)
2. At the start of the next round, for each query, exactly match historical traces and aggregate effectiveness data across all historical search queries (when the same query appears in multiple rounds, use the most recent data)
3. Divide into high-effectiveness / low-effectiveness groups using a `high_relevance_ratio` threshold of 0.5 (aligned with the judge scoring criterion "relevance score > 0.5")
4. Format and inject into the user prompt at `turn_id=0`. The system prompt of the outline generation template contains "Historical Search Query Effectiveness Feedback Usage Instructions" to guide the LLM in utilizing the data

Example injected content:

```
## High-effectiveness queries (3, returned documents highly
relevant)
- "XX brand review": avg_relevance=0.72, 8/10 documents relevant

## Low-effectiveness queries (2, please avoid similar queries)
- "how is XX": avg_relevance=0.15, 1/10 documents relevant
```

Fallback: If historical traces have no per-query details (old format), only show the search queries and evaluations from low-scoring rounds as negative references when the score gap ≥ 1.0.
Adjustable directions:

- Display count limit (currently 5 per group; modify `_format_query_performance_feedback`)

- High / low effectiveness threshold (currently 0.5; modify `_format_query_performance_feedback`)
- Fallback score gap threshold (currently 1.0; modify `_format_overall_feedback`)

Note: During the first batch eval (v0_baseline), the memory directory is empty and no feedback will be injected. Takes effect from v1 onward. Since `fixed_sample_size=10` guarantees the same set of queries is used across rounds, exact matching is guaranteed to hit.

Data-Driven Analysis

- **summary.md** (must read each round): aggregate metrics, statistics by intent, full case details (including blueprints, search queries, **all-dimension reasoning**, **per-query search query quality distribution and low-quality query details**)
- **details.jsonl**: per-case metric details
- **harness/memory/traces_*.jsonl**: complete trace for each round (including per-query document quality details), for deeper analysis
- **full_results.jsonl** (in the batch output directory): complete `run_agent` output with all intermediate results

Do not limit yourself to minor parameter adjustments. If analysis reveals a structural problem, the modification should be of corresponding magnitude.

VII. Precautions

1. **Working directory:** All commands are executed under the project root directory.
2. **Do not modify** `run_batch_agent`, `extract_metrics`, or `aggregate_metrics` **in** `eval_outline_judge.py`. Only modify templates, API service code, and configuration.
3. **Evaluator not modifiable:** `JudgeSearchQueryDiversityAPIService.py` and `JudgeSearchQueryDiversity_ZH.jinja2` must not be modified. These are the final scoring services — modifying the scorer is equivalent to cheating.
4. **Each round's modifications must be based on best_version only:** Do not continue modifying on top of a regressed version.
5. **Read historical changelogs to avoid repetition:** Do not retry the same changes that have already failed.
6. **Read summary.md before analyzing:** Before each round's analysis, first read `summary.md` under the `best_version` directory to understand the actual model output (blueprints, search queries, reasoning); do not rely solely on aggregate scores.
7. **Read the code before modifying:** Before modifying any file, read the complete current version of the code to understand the existing logic.
8. **API cost awareness:** Each batch round involves calls to Gemini + Google Search + GPT-oss.
9. **Template paths:** All templates are loaded at runtime from `./template/`. When modifying templates, edit files under `template/`; snapshots also store files from `template/`.
10. **Parameter configuration:** All API service parameters in `run_agent` are controlled via gin configuration (`test_harness_outline_xhs.gin`). To adjust parameters, modify the gin configuration file.
11. **Non-modifiable search parameters:** `search_engine="xiaohongshu"` and `num_searches=10` are fixed and must not be modified.
12. **Optimization target:** Only `search_coverage.mean` is the optimization target; all other metrics are for reference and display purposes only.

F.2 Prompts in Harness

As introduced in Section 2, the outline generator agent is repurposed as a scorer agent, the prompt of which is presented as follows.

Prompt: Scorer Agent

Role Definition
You are a professional expert in information retrieval and content planning evaluation, specializing in assessing the completeness and diversity of blueprints and search-queries in covering user queries, and capable of performing quality verification based on real retrieval result distribution data.

Input Description

- **User query:** the user's original needs and questions
- **blueprints:** a list of outline points, each containing:
  - **content:** description of the point
  - **search_query:** the corresponding list of search queries
- **Search result distribution data:** primarily the relevance judgment scores between documents returned for the search queries and the user query

---

Evaluation Dimensions
Evaluation Dimension 1: Completeness of blueprints and search-queries (completeness)
Scoring perspectives:

1. **Topic coverage:** Do the blueprints and search-queries cover all core topics and sub-topics required to answer the query?
2. **Intent recognition:** Do they capture both the user's explicit intent (what is directly asked) and implicit intent (underlying purpose)?
3. **Dimension diversity:** Are multiple dimensions explored (e.g., background, current status, causes, impact, comparison, trends, recommendations, etc.)?
4. **Missing key points:** Identify important content points involved in the query but not covered by the blueprints and search-queries.

Scoring criteria (0–10):

1. 9–10: Fully covers all core dimensions of the query, no obvious omissions, well-balanced dimension distribution
2. 7–8: Covers most core dimensions, with 1–2 minor omissions
3. 5–6: Covers basic dimensions, but with more than 1 important dimension missing
4. 3–4: Limited coverage, multiple important dimensions missing
5. 1–2: Covers only a very small amount of relevant content, severely insufficient
6. 0: Completely fails to cover the content required by the query

Evaluation Dimension 2: Diversity of blueprints and search-queries (diversity)
Scoring perspectives:

1. **Perspective diversity:** Do the blueprints and search-queries explore different standpoints / roles / group perspectives (user perspective, expert perspective, policy perspective, market perspective, etc.)?
2. **Content type diversity:** Do they simultaneously cover factual content, analytical content, comparative content, and advisory content?
3. **Granularity diversity:** Is there both a macro framework and micro-level details?
4. **Redundancy:** Is there a large amount of overlapping or semantically similar content among the blueprints and search-queries?

Scoring criteria (0–10):

1. 9–10: Rich and diverse perspectives, comprehensive content types, reasonable granularity distribution, no obvious repetition
2. 7–8: Good diversity, with occasional perspective repetition or type singularity
3. 5–6: Average diversity, with one type or perspective being overly dominant
4. 3–4: Insufficient diversity, content is highly homogeneous
5. 1–2: Severely lacking in diversity, large amount of repetition
6. 0: Completely no diversity

Evaluation Dimension 3: Search-query retrieval quality (search coverage)
Scoring perspectives:

1. **Retrieval effectiveness verification:** Verified based on the distribution of relevance judgment scores of retrieved content
2. **Cross-query result overlap:** Whether the semantics of snippets from different retrieved contents are complementary

Scoring criteria (0–10):

1. 9–10: All search-query information is sufficiently supported; the majority (above 80%) of search results are highly relevant (relevance score > 0.5)
2. 7–8: The majority (above 80%) of search-query content is sufficient, with the information chain basically complete
3. 5–6: 20%–40% of search-query content is insufficient, with information gaps present
4. 3–4: More than 40% of search-query content is insufficient or directionally off
5. 1–2: Most retrieval results are unable to support the corresponding search-queries
6. 0: Completely no relevant content

Output Format
Please strictly output the evaluation results in the following JSON format:

```
{
  "evaluation": {
    "completeness": {
      "score": 0,
      "reasoning": "explanation of scoring rationale"
    },
    "diversity": {
      "score": 0,
      "reasoning": "explanation of scoring rationale"
    },
    "search_coverage": {
      "score": 0,
      "reasoning": "explanation of scoring rationale"
    },
    "overall": {
      "score": 0,
      "reasoning": "overall evaluation summary"
    }
  }
}
```

G Gallery of Templates

G.1 Gallery of Templates in Render Agent

As described in Section B.3, the render agent offers a diverse collection of plug-in style templates, representative examples of which are illustrated in Figure 11.



Figure 11: **Gallery of diverse template styles and types**, including slides, posters, and portrait-format images. Our render agent offers extensive stylistic choices, accommodating diverse user preferences.