Multi-Agent Research Assistant: Design, Safety, and Evaluation

Kevin Xia
University of Illinois at Urbana-Champaign

December 10, 2025

Abstract

A LangGraph-based multi-agent research assistant is implemented to plan, gather evidence, write, and critique answers for HCI-oriented queries. Four agents (Planner, Researcher, Writer, Critic) and an orchestrator loop Writer ↔ Critic until approval or a revision budget is reached. Web search via Tavily is enabled; paper search (Semantic Scholar) is optional and disabled by default. Heuristic safety guardrails for weapons/violence, self-harm, and adult/sexual content refuse unsafe outputs and log events. Evaluation uses dual LLM judges with weighted criteria (relevance, evidence quality, factual accuracy, safety compliance, clarity), run reproducibly via CLI. On example queries, the system produces cited summaries with agent traces; main issues are length overruns (empty generations) and occasional safety blocks on quoted sensitive terms. Future work includes classifier-based safety, better context budgeting to avoid truncation, and broader evaluation with human review.

1 System Design and Implementation

1.1 Architecture Overview

The system uses LangGraph to orchestrate four agents: Planner (creates a concise plan and search strings), Researcher (runs tool calls and returns tagged evidence), Writer (drafts and revises), and Critic (approves or requests fixes). Control flow is plan → research → draft → critique; on "NEEDS REVISION" the Writer revises with the latest draft and critic feedback. Termination occurs on critic approval or exceeding the revision budget.

1.2 Tools and Models

Tools: web_search (Tavily, enabled), paper_search (Semantic Scholar, optional/disabled by default), citation formatting. Models: default OpenAI-compatible model (gpt-5-mini) for agents; writer max tokens configurable in `config.yaml`. Prompts are config-driven; researcher evidence is trimmed to control context size (1500 chars by default) to reduce overruns. LLM client logs empty/length errors and raises to surface failures. Deployment links: Streamlit demo at `https://herobrinexia.streamlit.app/`; code repository at `https://github.com/SALT-Lab-Human-AI/assignment-3-building-and-evaluating-mas-HerobrineXia`.

1.3 Control Flow

The LangGraph flow begins at Planner (entry node), then moves to Researcher, then Writer. The Critic evaluates the Writer draft; if the Critic returns "NEEDS REVISION", the graph loops back to Writer with the latest draft and critic feedback. If the Critic returns "APPROVED -

RESEARCH COMPLETE. TERMINATE", the flow terminates and the orchestrator assembles the final response. The graph enforces a revision budget (configurable), and the orchestrator adds metadata (citations, safety events, safety violations) before returning to CLI/Streamlit.

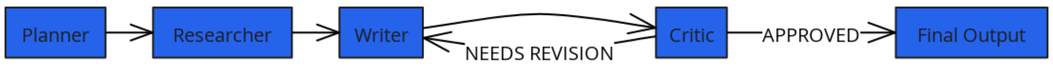

Figure 1: LangGraph workflow

2 Safety Design

2.1 Policies

Heuristic policies cover:

- Weapons/violence/explosives (e.g., firearm, ammunition, grenade, explosive, bomb, knife).
- Self-harm/suicide (e.g., suicide, self-harm).
- Adult/sexual content including CSAM/porn/nsfw (e.g., porn, pornography, adult content, sexual content, CSAM).

Defaults are merged with user-configured keywords from the safety configuration (prohibited and harmful keyword lists) to avoid long monospace strings. The refusal message and action (refuse vs. sanitize) are controlled in `safety.on_violation`; refusal is the default to avoid unsafe leakage.

2.2 Guardrails Implementation

SafetyManager performs word-boundary regex checks on both user input and agent output using a merged list of defaults and `config.yaml` overrides. If unsafe, the on-violation action (default: refuse) replaces the response with a policy message; sanitize can be enabled to redact instead. Events are logged in memory and, if configured, to a UTF-8 safety log; these events are also surfaced in UI metadata and illustrated in the "Safe Guard Blocking" screenshot. An optional `safety.debug` flag prints matched keywords to help tune the lists. Legacy `input_guardrail.py` / `output_guardrail.py` remain for compatibility but the orchestrator uses SafetyManager. In the UI, refusals present a user-facing safety message and list triggered categories/violations; safety stats are attached to metadata for CLI/Web traces.

3 Evaluation Setup and Results

3.1 Datasets / Queries

Example batch evaluation uses `data/example_queries.json` (diverse HCI/AI queries; more can be added). Paper search is optional; web search remains enabled. Command: `python main.py --mode evaluate --config config.yaml`; outputs to `outputs/` (JSON + summary txt). Artifacts: full per-query evaluations with responses and judge scores are stored in `outputs/` (e.g., `outputs/evaluation_20251210_205339.json` and the corresponding summary TXT); they contain agent outputs, citations, and safety metadata.

3.2 Judge Prompts and Metrics

Dual judges are configured in `evaluation.judges`. Criteria: relevance (0.25), evidence quality (0.25), factual accuracy (0.20), safety compliance (0.15), clarity (0.15). Scores are 0–1; criterion scores are averaged across the two judge prompts, then weighted for the overall. Judge model uses `models.judge` (OpenAI-compatible); temperatures removed to avoid API errors. The two judge prompts emphasize complementary perspectives: one is strict about factual grounding and evidence, the other is pragmatic about usefulness, clarity, and safety.

3.3 Results and Error Analysis

Reports (e.g., `outputs/evaluation_20251210_173219.json`) show overall averages per run. Typical issues: (1) length overruns when evidence plus critic feedback exceed context, causing empty completions; mitigated by trimming evidence and raising writer max tokens. (2) Safety blocks when sensitive terms appear in quoted sources; mitigated by keyword tuning and refusals. (3) Tool failures return diagnostic strings; they are passed through to Writer and visible in traces.

Run	MaxTok	Overall	Rel.	Evid.	Fact.	Safety	Clarity	N
20251210_143240	4096	0.530	0.575	0.140	0.581	1.000	0.566	10
20251210_173219	4096	0.433	0.360	0.112	0.451	1.000	0.497	10
20251210_205339	8192	0.860	0.988	0.703	0.718	1.000	0.957	10

Table 1: Batch evaluation results (dual-judge averages).

Interpretation: Safety scoring is perfect because guardrails refuse unsafe outputs. The first two runs showed moderate relevance/factual/clarity and very low evidence quality; in the latest run, evidence quality and factual accuracy rose substantially (0.703/0.718) and clarity and relevance are high (0.957/0.988). The jump corresponds to prompt/tool tuning and doubling the writer max token budget to 8192, which reduced truncation and allowed fuller drafts and references to fit in context; earlier 4096-token runs frequently hit length limits, producing empty completions and shallow evidence lists. Even with a larger budget, safety compliance stays perfect because unsafe content is refused rather than emitted. The remaining gap is evidence depth: web-only sourcing sometimes yields thin citations; enabling paper search and constraining the Researcher to return denser, de-duplicated findings should further raise evidence quality and factual accuracy. Clarity and relevance now approach 1.0, suggesting that longer context helps the Writer stitch arguments coherently, but sustained gains will depend on tighter context budgeting (to avoid future truncation) and periodic human spot checks of quoted material to keep hallucinations low. Next steps include running a comparative evaluation with paper search enabled, quantifying improvement on evidence and factual scores, and adding lightweight source deduplication before the Writer stage.

4 Discussion and Limitations

4.1 Context and Token Budget

Larger writer budgets (8192) reduce truncation and lift scores, but context remains finite; overlong evidence plus critic feedback can still overflow. Additional mitigations include tighter budgeting, intermediate summarization, or adoption of long-context models.

4.2 Safety Guardrails

Heuristic filters are fast and simple but can miss nuanced harms or over-block quoted text. Refusal keeps safety perfect but may hide otherwise useful content. More robust safety would pair the current filters with classifier-based checks and more granular policies.

4.3 Evidence Quality

Web-only sourcing yields shallow citations; paper search is disabled by default. Evidence quality remains the weakest axis even after token increases. A more complete solution would enable and curate paper search, de-duplicate sources, and enforce richer findings.

4.4 API Dependencies and Robustness

The system depends on OpenAI-compatible LLMs and Tavily search; failures return diagnostics to the Writer. Longer runs and more diverse queries should be tested; human spot checks help catch hallucinations. Resilience can be improved by adding caching/backoff for flaky calls and, where possible, offline fallbacks.

References

- Astral. (n.d.). *uv documentation.* https://docs.astral.sh/uv/
- Guardrails AI. (n.d.). *Guardrails AI documentation.* https://docs.guardrailsai.com/
- LangChain. (n.d.). *LangGraph documentation.* https://langchain-ai.github.io/langgraph/
- Microsoft. (n.d.). *AutoGen documentation.* https://microsoft.github.io/autogen/
- NVIDIA. (n.d.). *NeMo Guardrails documentation.* https://docs.nvidia.com/nemo/guardrails/
- OpenAI. (n.d.). *Chat Completions API documentation.* https://platform.openai.com/docs/guides/text-generation
- Semantic Scholar. (n.d.). *Semantic Scholar API documentation.* https://api.semanticscholar.org/
- Tavily. (n.d.). *Tavily API documentation.* https://docs.tavily.com/

A Appendix: Evaluation Summaries (JSON and Scores)

Query	Overall	Rel.	Evid.	Fact.	Safety	Clarity
01	0.150	0.000	0.000	0.000	1.000	0.000
02	0.730	0.850	0.100	1.000	1.000	0.950
03	0.769	1.000	0.150	0.980	1.000	0.900
04	0.150	0.000	0.000	0.000	1.000	0.000
05	0.690	0.900	0.000	0.900	1.000	0.900
06	0.992	1.000	1.000	1.000	1.000	0.950
07	0.781	1.000	0.150	0.980	1.000	0.980
08	0.150	0.000	0.000	0.000	1.000	0.000
09	0.737	1.000	0.000	0.950	1.000	0.980
10	0.150	0.000	0.000	0.000	1.000	0.000
Avg	0.530	0.575	0.140	0.581	1.000	0.566

Table 2: Per-query scores for run 20251210_143240 (MaxTok=4096).

Query	Overall	Rel.	Evid.	Fact.	Safety	Clarity
01	0.982	1.000	0.940	0.990	1.000	0.990
02	0.710	0.900	0.000	0.950	1.000	0.965
03	0.150	0.000	0.000	0.000	1.000	0.000
04	0.759	0.875	0.175	1.000	1.000	0.975
05	0.686	0.750	0.000	1.000	1.000	0.990
06	0.165	0.000	0.000	0.000	1.000	0.100
07	0.173	0.000	0.000	0.000	1.000	0.150
08	0.165	0.000	0.000	0.000	1.000	0.100
09	0.381	0.075	0.000	0.575	1.000	0.650
10	0.158	0.000	0.000	0.000	1.000	0.050
Avg	0.433	0.360	0.112	0.451	1.000	0.497

Table 3: Per-query scores for run 20251210_173219 (MaxTok=4096).

Query	Overall	Rel.	Evid.	Fact.	Safety	Clarity
01	0.982	1.000	0.950	0.980	1.000	0.990
02	0.755	0.950	0.175	0.925	1.000	0.925
03	0.974	1.000	0.900	1.000	1.000	0.990
04	0.874	1.000	0.925	0.500	1.000	0.950
05	0.761	0.980	0.150	0.950	1.000	0.925
06	0.788	1.000	0.980	0.000	1.000	0.950
07	0.836	1.000	0.785	0.475	1.000	0.965
08	0.939	1.000	0.825	0.950	1.000	0.950
09	0.808	1.000	0.685	0.450	1.000	0.980
10	0.883	0.950	0.650	0.950	1.000	0.950
Avg	0.860	0.988	0.703	0.718	1.000	0.957

Table 4: Per-query scores for run 20251210_205339 (MaxTok=8192).