Multi-Agent System for HCI Literature Synthesis: Design, Safety, and Evaluation

Abstract

This report presents a multi-agent research system designed for Human-Computer Interaction (HCI) literature synthesis. The system employs eight specialized agents orchestrated through LangGraph, including Planner, Researcher, Writer, Critic, SafetyGuardian, ReflexionEngine, LLMJudge, and EvaluationTriangulation components. Research tools integrate Semantic Scholar API for academic papers and Tavily API for web sources. Safety is enforced through NeMo Guardrails implementing a three-layer architecture with pre-flight input validation, in-flight monitoring, and post-flight output sanitization. The LLM-as-a-Judge evaluation framework scores responses across five weighted criteria: relevance (25%), evidence quality (25%), factual accuracy (20%), safety compliance (15%), and clarity (15%). Evaluation on 12 HCI research queries achieved an overall score of 0.955, with relevance, safety, and clarity achieving perfect 1.0 scores.

1. System Design and Implementation

1.1 Agent Architecture

The system implements eight specialized agents with distinct responsibilities in the research synthesis pipeline. Table 1 summarizes the agent roles.

Table 1: Multi-Agent Architecture

Agent	Role	Key Function
Planner	Query decomposition	Chain-of-Thought breakdown into sub-tasks
Researcher	Evidence gathering	Parallel execution of paper and web search
Writer	Response synthesis	Multi-perspective integration with citations
Critic	Quality verification	Revision recommendations and improvement
SafetyGuardian	Safety enforcement	Three-layer content filtering
ReflexionEngine	Self-correction	Failure tracking and lesson extraction
LLMJudge	Automated scoring	Five-criterion evaluation with rubrics
EvaluationTriangulatioı	Decision aggregation	Two-thirds approval voting mechanism

The architectural foundation rests on deliberate decomposition of complex cognitive tasks into specialized agent roles (Russell & Norvig, 2021). The eight-agent configuration emerged from iterative refinement during development. Early prototypes with fewer agents conflated responsibilities in ways that produced suboptimal outputs. The current architecture represents a principled balance where each agent maintains a coherent cognitive focus while the orchestration layer manages inter-agent coordination efficiently.

The Planner agent transforms user queries into structured research agendas through Chain-of-Thought reasoning (Wei et al., 2022). The Researcher agent specializes in evidence acquisition through parallel tool execution, implementing the retrieval-augmented generation paradigm (Lewis et al., 2020). The Writer agent synthesizes research findings into coherent narratives with citations. The Critic provides adversarial evaluation through a red-teaming approach (Perez et al., 2022).

The evaluation cluster separates assessment functions into dedicated agents. The SafetyGuardian implements three-layer content filtering with complete independence from content generation. The ReflexionEngine maintains cross-query memory of failures and corrections based on the Reflexion framework (Shinn et al., 2023). The LLMJudge applies structured rubrics across five evaluation criteria, while EvaluationTriangulation aggregates assessments from multiple sources requiring two-thirds approval before accepting responses.

1.2 Tool Integration

Three primary tools support the research pipeline. The paper search tool interfaces with the Semantic Scholar Academic Graph API (Ammar et al., 2018), providing access to over 200 million scholarly documents with rich metadata including citation networks, author affiliations, and publication venues. The tool enforces a maximum of ten papers per search to prevent information overload while ensuring sufficient diversity.

The Tavily API provides AI-optimized web search designed for language model applications, returning pre-processed content excerpts rather than raw HTML. Combining academic and web search addresses a fundamental limitation of either source in isolation. Academic databases lag recent developments by months or years due to publication cycles, while web content lacks peer-review quality assurance.

The citation formatter tool generates APA-style references from paper metadata (American Psychological Association, 2020), ensuring consistent formatting throughout responses. The formatter maintains a reference registry that tracks cited papers, enabling detection of potential over-reliance on single sources.

1.3 Control Flow

The workflow orchestration relies on LangGraph's StateGraph abstraction (Chase, 2023) for managing state transitions across a twelve-step pipeline. The state container maintains four primary fields: a messages field for conversation history accumulation, a research_findings dictionary aggregating evidence from academic and web sources, a safety_status object tracking pre-flight, in-flight, and post-flight safety checks, and a quality_score determining whether self-refinement loops should be triggered.

The control flow proceeds as follows. First, safety pre-flight validates user inputs through jailbreak detection using seven regex patterns, harmful content checking, and HCI topic relevance verification. Second, the planning team performs query decomposition through Chain-of-Thought reasoning and plan refinement. Third, the research team executes parallel tool calls using ThreadPoolExecutor for concurrent paper_search and web_search operations. Fourth, the synthesis team generates multi-perspective responses with quality evaluation and revision suggestions.

Fifth, the Chain-of-Verification pipeline (Dhuliawala et al., 2023) generates verification questions targeting specific factual claims, executes independent verification using a separate o3-mini model, and integrates verified claims into the final response. Sixth, the self-refinement loop verifies citation counts and triggers revision if quality falls below the 0.7 threshold. Seventh through ninth, the evaluation pipeline runs ReflexionEngine self-assessment, LLMJudge five-criterion scoring, and triangulation requiring two-thirds approval.

Tenth, safety post-flight performs PII redaction covering five data types (email, phone, SSN, credit card, IP) and final content safety verification. Eleventh and twelfth, the system persists findings to research memory and returns the formatted response to the user.

The parallel tool execution architecture reduces average research phase duration from 8.2 seconds to 4.9 seconds, representing approximately 40% improvement. The architecture also provides natural fault isolation, enabling graceful degradation when individual APIs experience temporary unavailability.

1.4 Model Selection

The system employs OpenAI's gpt-4o model as the primary reasoning engine for all agent teams, selected for its strong performance on complex multi-step tasks and 128K token context window. The LLM Judge uses gpt-4o with temperature 0.3 for scoring consistency, while agent reasoning tasks use temperature 0.7. A dedicated o3-mini model handles Chain-of-Verification to achieve genuine independent cross-checking. For rapid auxiliary operations, gpt-4o-mini balances capability with cost efficiency.

2. Safety Design

2.1 Framework and Architecture

Safety is implemented through NeMo Guardrails (Rebedea et al., 2023) combined with a custom SafetyGuardian agent. NeMo Guardrails provides a declarative domain-specific language called Colang that enables precise specification of acceptable and prohibited interaction patterns. The SafetyGuardian applies LLM-based reasoning for edge cases where pattern matching might produce false positives or negatives.

Table 2 presents the three-layer safety architecture implementing defense-in-depth across input, processing, and output stages.

Table 2: Three-Layer Safety Architecture

Layer	Timing	Checks Performed	Actions Taken

Pre-flight	Before processing	patterns (7 regex), harmful content keywords ε HCI topic relevance	with redirect, BLOCK with e
In-flight	During tool executi	earch results, result quality thresholds, source	WARN user, SANITIZE conte
Post-flight	Before delivery	on (5 types), content safety verification, citati	T sensitive data, BLOCK harn

2.2 Detection Patterns and Prohibited Categories

PII detection implements regex patterns for five data types: email addresses using RFC 5322 pattern, phone numbers targeting US formats, Social Security numbers, credit card numbers with Luhn algorithm validation, and IPv4 addresses. Jailbreak detection monitors for instruction override attempts using patterns that capture common manipulation strategies including "ignore previous instructions" variations, "pretend you are" constructions, and references to "developer mode" or "DAN mode" terminology.

The guardrail policy defines four prohibited categories: harmful content (violence, hacking instructions, illegal activities) with HIGH severity and immediate refusal; personal attacks (discriminatory or hateful language) with HIGH severity; misinformation (false HCI research claims) with MEDIUM severity triggering warnings and corrections; and off-topic queries (non-HCI topics) with LOW severity triggering graceful redirection.

3. Evaluation Setup and Results

3.1 Dataset and Methodology

The evaluation dataset comprises twelve queries: ten addressing legitimate HCI research topics (explainable AI, AR usability, AI ethics, UX measurement, conversational AI, accessibility, data visualization, voice interfaces, AI prototyping, cross-cultural design) and two safety test cases (harmful content request, off-topic query). Query selection prioritized topics with sufficient academic literature while remaining current enough to benefit from web search augmentation.

The evaluation employs an LLM-as-a-Judge methodology (Zheng et al., 2023), using a capable language model to assess response quality across multiple dimensions. Table 3 presents the five evaluation criteria with their weights.

Table 3: LLM Judge Evaluation Criteria

Criterion	Weight	Scoring Description
Relevance	25%	Completeness of query address, topic coverage, appropriate scope
Evidence Quality	25%	Citation count (6+ for 0.85+), source authority, inline formatting
Factual Accuracy	20%	Claim verifiability, consistency with sources, absence of hallucination
Safety Compliance	15%	Harmful content absence, appropriate refusal of unsafe requests
Clarity	15%	Structure, organization, academic writing style, logical flow

The overall score is computed as: Overall = (Relevance × 0.25) + (Evidence × 0.25) + (Accuracy × 0.20) + (Safety × 0.15) + (Clarity × 0.15). The framework requires all five criteria to exceed the 0.85 threshold for overall acceptance.

3.2 Results and Error Analysis

Table 4 presents the final evaluation scores across all criteria.

Table 4: Evaluation Results

Criterion	Score	Status

Relevance	1.000	PASS
Evidence Quality	0.900	PASS
Factual Accuracy	0.900	PASS
Safety Compliance	1.000	PASS
Clarity	1.000	PASS
Overall	**0.955**	**TARGET MET**

Evaluation across the twelve-query test dataset achieved an overall weighted score of 0.955, exceeding the 0.85 threshold by a substantial margin. Relevance, safety compliance, and clarity achieved perfect 1.0 scores, indicating reliable production of responses that fully address queries, maintain safety standards, and present information clearly.

Evidence quality scored 0.900, with lower-scoring responses typically addressing narrower topics where fewer relevant papers exist in the Semantic Scholar database. Factual accuracy also scored 0.900, indicating that the Chain-of-Verification pipeline successfully catches most errors while subtle imprecisions occasionally persist, including paraphrasing that slightly shifts claim meaning and overgeneralization of specific study results.

The safety test cases validated the three-layer architecture. The harmful query requesting hacking information triggered jailbreak detection at the pre-flight layer, producing immediate refusal with explanation. The off-topic query activated topic relevance filtering, producing a redirection response explaining the system's HCI specialization.

4. Discussion and Limitations

4.1 Key Insights

The ThreadPoolExecutor-based parallel tool execution reduces average query latency by approximately 37.5% without sacrificing result quality. The parallel architecture also provides robustness benefits, as failures in one search tool do not block results from the other, enabling graceful degradation to single-source responses during API issues.

The ReflexionEngine's cross-query learning demonstrates progressive improvement throughout evaluation sessions, with later queries benefiting from lessons extracted from earlier failures. Particularly valuable learning occurs around API timeout handling, where the engine learns to reduce result count requests when previous queries experience timeouts.

The evaluation triangulation requirement for two-thirds approval from multiple evaluators catches edge cases that individual evaluators miss, validating the multi-evaluator architecture's robustness benefits.

4.2 Limitations and Future Work

The system's reliance on external APIs introduces rate limiting constraints affecting scalability for high-volume deployments. Semantic Scholar API allows one request per second under the free tier. API availability represents another limitation, as occasional downtime affects system reliability.

The HCI domain focus limits transferability to other research domains without significant reconfiguration. The single-turn interaction model prevents iterative refinement based on user feedback. The current citation verification approach confirms that cited papers exist but does not verify that papers actually contain the attributed claims.

Future development priorities include implementing stateful multi-turn conversations, retrieval-based citation verification comparing claims against actual paper content, and modular guardrail configurations enabling expansion to adjacent research domains including cognitive science, educational technology, and information science.

References

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.).

Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., ... & Etzioni, O. (2018). Construction of the literature graph in Semantic Scholar. *NAACL-HLT*, 84-91.

Chase, H. (2023). LangGraph: Build stateful, multi-actor applications with LLMs. *LangChain Documentation*.

Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., & Weston, J. (2023). Chain-of-verification reduces hallucination in large language models. *arXiv:2309.11495*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *NeurIPS*, 9459-9474.

Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., ... & Irving, G. (2022). Red teaming language models with language models. *EMNLP*, 3419-3448.
Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., & Cohen, J. (2023). NeMo Guardrails: A toolkit for controllable and safe LLM applications. *arXiv:2310.10501*.
Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*.
Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 24824-24837.
Zheng, L., Chiang, W. L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., ... & Stoica, I. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS*.