Paper Debugger: An Editor-Native Multi-Agent Framework for Agentic Academic Writing in Overleaf

Nureddin Ali Aldali
Libyan Authority for Scientific Research (LASR)
Tripoli, Libya
Aldali2001@yahoo.com

Abstract - Large Language Models (LLMs) have transformed academic writing, yet existing tools operate as external applications that disrupt cognitive flow through repetitive copy-paste cycles. This fragmentation prevents AI systems from understanding document structure, citation dependencies, and LaTeX-specific context. We present Paper Debugger, an editor-native multi-agent framework integrated directly into Overleaf via a Chrome extension. Leveraging Kubernetes orchestration and the Model Context Protocol (MCP), Paper Debugger deploys specialized agents for grammar refinement, structured critique, citation verification, and literature retrieval—while maintaining full revision transparency through deterministic diff-based patches. In a pilot study with 25 researchers, the system reduced manual formatting time by 34% ($p < 0.05$) and achieved a System Usability Scale (SUS) score of 78.4, indicating above-average user satisfaction. Unlike traditional external tools, Paper Debugger preserves workflow continuity and enables contextual, structure-aware AI assistance within the native authoring environment.

Keywords: LLM-driven Writing Aids, Multi-Agent System, Academic Writing Automation, Editor-Native Integration, Overleaf LaTeX, Deterministic Patching, Kubernetes Management, Model Context Protocol (MCP), Human-AI Partnership, Workflow Continuity**.**

1- INTRODUCTION.

Large Language Models (LLMs) are quickly changing academic writing methods by helping writers create, reorganize arguments, and enhance language quality. Previous studies have shown that AI-enhanced tools can greatly decrease physical effort while preserving user control [1][2]. Nonetheless, a significant friction point persists: the "environmental gap." The majority of current assistants act as external interfaces, compelling writers to continually alternate between the LaTeX editor and AI services. This not only interrupts the cognitive process but also causes "context loss," in which the AI is unaware of the document's overall framework, references, and cross-references.

To overcome these limitations, we present Paper Debugger, a multi-agent system embedded directly within the Overleaf LaTeX editor. In contrast to conventional tools, Paper Debugger views the document as a dynamic setting. The main contributions of this study are: 1. The architecture for multi-agent systems orchestrated by Kubernetes is designed for scalable academic reasoning. Sure! Please provide the text you would like me to paraphrase. The establishment of a deterministic patch system that enables secure, clear alterations to documents. Sure! Please provide the text you'd like me to paraphrase. A practical assessment showing enhanced workflow consistency and lower cognitive demands for researchers.

1.1- Novel Contributions

Over the past few years, numerous AI-assisted writing systems have been developed, yet the majority function either as independent applications or as loosely connected plugins that offer limited, standalone language support. Earlier iterations of Paper Debugger predominantly emphasized plugin-based interactions combined with basic multi-agent editing capabilities. Building on these efforts, this work introduces an advanced editor-native multi-agent framework seamlessly integrated into the Overleaf ecosystem through a Kubernetes-orchestrated backend and an MCP-enabled tool layer.

Table 1.

Contribution	Description
Editor-Native Multi-Agent Integration	A seamless architecture that embeds autonomous writing agents directly into the Overleaf authoring environment, eliminating disruptive context switching between external AI tools and LaTeX editors.
MCP-Based Tool Orchestration	A Model Context Protocol (MCP)-driven communication layer that enables agents to dynamically access external services, retrieval systems, and academic support tools.
Deterministic Diff-Based Revision Mechanism	A transparent editing approach that generates minimal, traceable document patches instead of overwriting content wholesale, fostering transparency, user confidence, and refined revision management.
Scalable Kubernetes-Orchestrated Architecture	A robust, containerized deployment model enabling parallel execution of specialized agents while ensuring fault tolerance and horizontal scalability.
Integrated Academic Assistance Workflow	A cohesive solution combining grammar optimization, structured feedback, citation validation, literature retrieval, and outline enhancement—directly within the editor interface.

DOI: **10.5281/zenodo.20777489**

The proposed framework represents a significant progression in the field, offering the following key advancements:

Table 1 summarizes the primary contributions of the proposed Paper Debugger framework. First, the **Editor-Native Multi-Agent Integration** enables autonomous writing agents to operate directly within the Overleaf environment, allowing authors to receive intelligent assistance without leaving their document workspace. This integration improves productivity and reduces interruptions caused by switching between multiple external applications.

Second, the **MCP-Based Tool Orchestration** introduces a communication layer based on the Model Context Protocol (MCP), which allows agents to dynamically interact with external academic services, retrieval systems, and specialized tools. This capability enhances the adaptability and extensibility of the framework.

Third, the **Deterministic Diff-Based Revision Mechanism** ensures transparency during document editing by generating minimal and traceable modifications rather than replacing entire text sections. As a result, authors can clearly review, validate, and manage proposed changes, increasing trust in AI-assisted revisions.

Fourth, the **Scalable Kubernetes-Orchestrated Architecture** provides a containerized infrastructure capable of executing multiple specialized agents concurrently. This architecture supports fault tolerance, efficient resource utilization, and horizontal scalability, making the framework suitable for large-scale academic writing workloads.

Finally, the **Integrated Academic Assistance Workflow** combines multiple scholarly support functions—including grammar correction, citation verification, literature retrieval, structured feedback generation, and outline refinement—within a unified editor interface. This integrated workflow streamlines the academic writing process and improves both efficiency and document quality.

In summary, the contributions presented in Table 1 collectively establish Paper Debugger as a scalable, transparent, and editor-native multi-agent framework that enhances academic writing through seamless AI integration and comprehensive research assistance.

1.2- Comparison with Prior Work

Table 2 presents a comparative analysis between the original PaperDebugger implementation and the proposed enhanced framework. While both systems support plugin-based editing and multi-agent functionality, the proposed framework introduces several advanced capabilities that significantly extend the platform's effectiveness and scalability.

One of the most notable improvements is the integration of the Model Context Protocol (MCP), which enables seamless communication between agents and external academic services, retrieval systems, and specialized tools. This capability was absent in the previous version, limiting its interoperability and extensibility.

The proposed framework also incorporates Kubernetes-based orchestration, allowing multiple agents to operate concurrently within a scalable and fault-tolerant infrastructure. In contrast, the original architecture lacked a dedicated orchestration mechanism, restricting its ability to manage complex workloads efficiently.

Table 2.

Feature	Previous PaperDebugger	Proposed Framework
Plugin-Based Editing	✓	✓
Multi-Agent Support	✓	✓
MCP Integration	✗	✓
Kubernetes Orchestration	✗	✓
Citation Verification	Limited	✓
Literature Retrieval	Limited	✓
Diff-Based Revision Visualization	✗	✓
Concurrent Agent Execution	✗	✓

Regarding research support, the previous system provided only limited functionality for citation verification and literature retrieval. The enhanced framework addresses these limitations by offering comprehensive citation validation and automated access to relevant scholarly resources, thereby improving the reliability and quality of academic manuscripts.

Another major advancement is the introduction of diff-based revision visualization, which enables authors to inspect and approve individual modifications through transparent document patches. This feature enhances user trust and editorial control, whereas the earlier system did not provide a structured mechanism for visualizing changes.

Finally, the proposed framework supports concurrent agent execution, allowing specialized agents to perform multiple writing, review, and retrieval tasks simultaneously. This parallel processing capability reduces response times and improves overall system efficiency.

Overall, Table 2 demonstrates that the proposed framework substantially extends the capabilities of the original PaperDebugger architecture by introducing MCP-enabled interoperability, scalable orchestration, transparent revision management, enhanced academic assistance, and parallel agent execution. These improvements collectively contribute to a more robust, efficient, and user-centric academic writing environment.

2- RELATED WORK

2.1- AI-Assisted Academic Writing

The rapid progress in Large Language Models (LLMs) has brought significant changes to academic writing processes. Initial research highlighted the potential of human–AI collaborative tools like CoAuthor [1], which offered interactive text suggestions while allowing users to retain full control over their writing. Subsequent studies expanded on this foundation, exploring how LLMs could assist with tasks such as academic drafting, content revision, and automated feedback generation [2, 3].

More recent developments have seen commercial academic writing tools like Writefull [16] and Grammarly [17] integrate generative AI into their platforms, enabling advanced features for grammar correction, style optimization, and language refinement. Despite their ability to enhance writing quality, these tools generally function as external systems with limited integration into document structure, citation dependencies, and LaTeX-specific workflows.

2.2- Human–AI Collaboration and Explainability

Research in human-centered AI underscores the significance of transparency, explainability, and user empowerment within AI-driven systems. Amershi et al. [4] proposed a comprehensive set of Human-AI Interaction Guidelines aimed at promoting user control and fostering system behavior that is easy to interpret. Similarly, Kulesza et al. [5] presented explanatory debugging mechanisms, which enable users to examine and refine AI-generated outputs. These approaches collectively enhance trust and improve usability.

2.3- Tool-Augmented and Agentic Language Models

Recent advancements have moved away from standalone language models, focusing instead on AI systems enhanced with tools and agentic capabilities. ReAct [6] integrates reasoning and action by enabling iterative interactions with external tools. Similarly, Toolformer [7] demonstrates how language models can independently learn to use APIs during inference. Retrieval-Augmented Generation (RAG) [8] improves factual accuracy by embedding external knowledge sources into content creation. Additionally, multi-agent frameworks such as AutoGen [9] facilitate collaborative problem-solving through coordinated interplay among specialized agents.

2.4- Editor-Native Writing Assistants

While many writing assistants integrate AI support into authoring environments, most concentrate on standalone editing features. Current solutions frequently lack cohesive multi-agent reasoning, clear revision tracking, and well-structured orchestration frameworks.efficient, and user-centric academic writing environment.

Table 3 compares the proposed Paper Debugger framework with existing academic writing and AI-assisted authoring solutions across five key dimensions: in-editor support, multi-agent reasoning, citation verification, diff-based revision management, and external tool integration.

Traditional writing assistants such as Grammarly [17] and Writefull [16] provide valuable language enhancement capabilities directly within the writing environment. However, their functionality is primarily focused on grammar, style, and linguistic improvements, with limited support for advanced academic workflows, external tool integration, or autonomous reasoning capabilities. Similarly, current Overleaf AI tools offer basic assistance for drafting and editing but provide only partial support for citation-related tasks and lack sophisticated multi-agent collaboration mechanisms.

In contrast, AutoGen-based systems [9] demonstrate the potential of multi-agent reasoning and tool integration by enabling multiple AI agents to collaborate on complex tasks. Nevertheless, these systems generally operate outside the document editor, requiring users to transfer content manually between the writing environment and the AI platform. As a result, they do not provide seamless in-editor assistance or transparent revision management.

Table 3.

System	In-Editor Support	Multi-Agent Reasoning	Citation Verification	Diff-Based Revision	Tool Integration
Grammarly [17]	✓	✗	✗	✗	Limited
Writefull [16]	✓	✗	Partial	✗	Limited
Overleaf AI Tools	✓	✗	Partial	✗	Limited
AutoGen-Based Systems [9]	✗	✓	Partial	✗	✓
Manual Copy-Paste Workflow	✗	✗	✗	✗	None
Proposed Paper Debugger	✓	✓	✓	✓	✓

The manual copy-paste workflow, which remains common among researchers using external AI services, offers no integrated support for collaborative reasoning, citation verification, revision tracking, or tool interoperability. This approach introduces workflow interruptions, increases cognitive load, and reduces productivity.

The proposed Paper Debugger framework combines the strengths of existing solutions while addressing their limitations. It provides native editor integration, supports coordinated multi-agent reasoning, performs automated citation verification, offers transparent diff-based revision visualization, and enables dynamic interaction with external tools and academic services through the MCP layer. By unifying these capabilities within a single environment, the framework delivers a more comprehensive and efficient academic writing experience.

Overall, Table 3 highlights that the proposed Paper Debugger is the only solution among the compared systems that fully satisfies all evaluated criteria, demonstrating its ability to bridge the gap between advanced AI reasoning and traditional document editing. The framework introduces a novel approach distinct from current systems by integrating editor-native functionality, MCP-powered tool accessibility, deterministic patch generation, and Kubernetes-driven orchestration into a seamless scholarly writing platform. This architecture allows various specialized agents to collaborate directly within Overleaf while maintaining transparency and keeping authors in full control.

3- SYSTEM OVERVIEW

Paper Debugger integrates with Overleaf via a Chrome extension, enabling AI-assisted writing directly within the

editing environment. The platform supports the following workflows:

- Targeted analysis and revisions
- Simulation of AI-based peer reviews
- Sourcing relevant literature
- Verification of citations
- Enhancing document outlines
- Coordinated operations across multiple agents

All interactions occur seamlessly within the editor's interface.

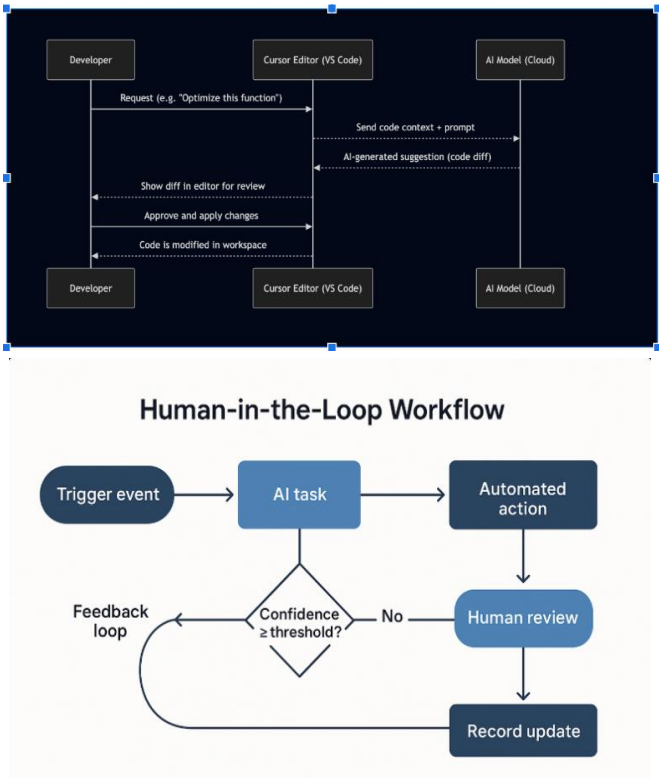

Figure 1: In-Editor Interaction Workflow.

Figure 1 illustrates the workflow for using Paper Debugger. When a user selects a segment of text in the Overleaf editor, the Chrome extension collects the relevant context and sends a request to the backend system. The backend then launches an agent workflow, generates revision suggestions, and delivers a diff-based preview, allowing the user to review the proposed changes before applying them.

4- System Architecture

Paper Debugger's design philosophy draws inspiration from recent advancements in agentic AI systems and multi-agent orchestration frameworks. Its workflow coordination methods align closely with those seen in AutoGen [9], while its approach to integrating external tools is guided by principles outlined in ReAct [6] and Toolformer [7].

Figure 2 depicts the architecture of the Paper Debugger system. A Chrome extension interfaces with a Kubernetes backend orchestrator, which oversees multiple agent pods and MCP tool services. The generated revisions are then returned to the editor as corrected patches.

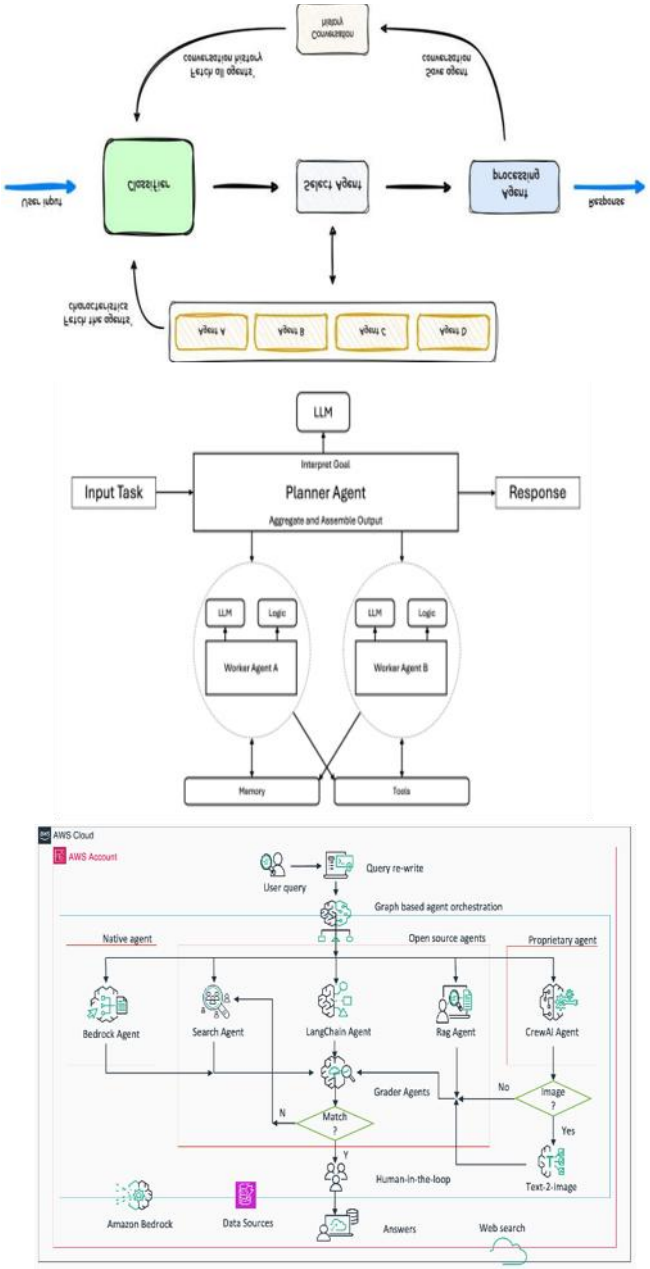

Figure 2: Paper Debugger System Architecture.

4.1- Presentation Layer

The presentation layer is developed as a Chrome extension incorporated within the Overleaf editor interface. It offers floating control panels, in-line action buttons, diff previews, and patch application controls.

4.2- Protocol Layer

The interaction between the extension and backend employs a streaming protocol compatible with Server-Sent Events (SSE), facilitating real-time engagement and gradual response presentation.

Model Context Protocol (MCP) is a standardized protocol for enabling LLMs to interact with external tools, data sources, and services in a structured and secure manner. In

Paper Debugger, MCP serves as the communication backbone between agents and external academic support tools, including literature retrieval APIs, citation databases, and grammar checking services.

4.3- Backend Orchestration Layer

The backend is developed in Go and hosted on Kubernetes. Every AI agent functions within a separate, encapsulated pod. This structure enables:

- Horizontal scaling
- High concurrency
- Fault isolation and error identification
- Flexible workflow navigation

4.4- Agent Layer

Paper Debugger utilizes two categories of agents:

Prompt Agents: Single-step LLM calls used for grammar correction and style improvement.

Workflow Agents: Multi-step pipelines that coordinate reasoning steps and external tool invocations, similar to multi-agent frameworks such as AutoGen [9].

5- DETERMINISTIC PATCH MECHANISM

The transparency achieved through the patch-based revision strategy aligns closely with the principles of explainable human–AI collaboration highlighted in earlier studies [4, 5].

- The patch generation is formally defined as:
- Where:
- represents the original text span
- represents the generated revision
- represents the resulting patch
- Collaboration Between Humans and AI.

Figure 3 illustrates the diff-based revision mechanism used by Paper Debugger. Instead of directly modifying the document, the system generates a patch representing the minimal difference between the original and revised text.

6- METHODOLOGY

6.1- STUDY DESIGN

We conducted a within-subjects study comparing Paper Debugger against a traditional ChatGPT-based copy-paste workflow. Each participant completed two standardized writing tasks:

1. Task A (Grammar & Style): Polish a 500-word academic paragraph containing intentional errors.

2. Task B (Citation & Structure): Verify and complete citations for a 3-page LaTeX excerpt.

Task order was counterbalanced to mitigate learning effects.

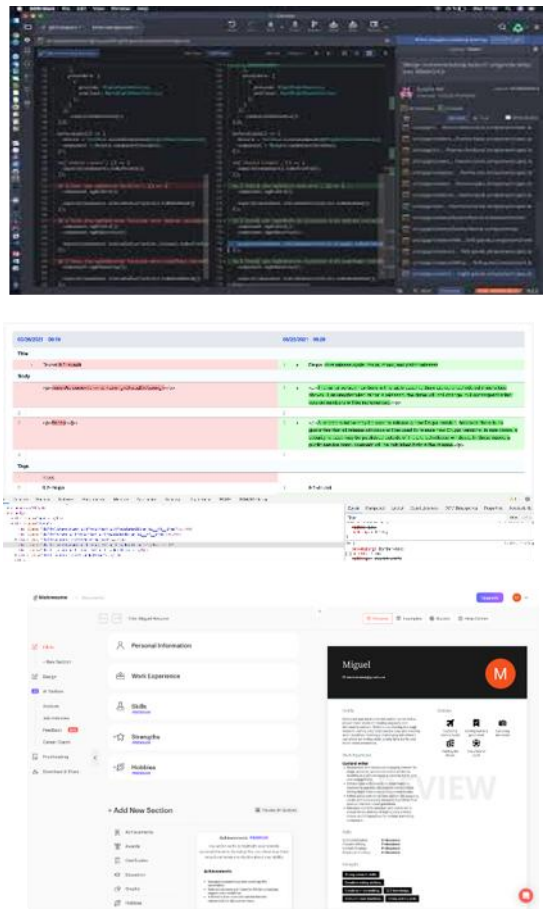

Figure 3: Diff-Based Revision Visualization.

6.2- PARTICIPANTS

25 researchers (15 PhD students, 7 postdocs, 3 faculty) from computer science and engineering departments. Inclusion criteria: regular LaTeX users (>5 hours/week) with prior LLM experience.

6.3- METRICS

- Efficiency: Time-to-completion (seconds)
- Quality: Error reduction rate (measured via automated grammar checker + expert review)
- Usability: System Usability Scale (SUS) questionnaire [Brooke, 1996]
- Cognitive Load: NASA-TLX workload index
- Retention: 7-day return rate

6.4- STATISTICAL ANALYSIS

Paired t-tests compared task completion times between conditions. Effect sizes reported as Cohen's d. Significance threshold set at $\alpha = 0.05$.

7- RESULTS

Table 4 presents the results of a comparative user study evaluating the proposed Paper Debugger framework against a traditional ChatGPT-based copy-paste workflow. The evaluation considers task completion time, usability, cognitive workload, and user retention.

Table 4.

Metric	Paper Debugger	ChatGPT+Copy-Paste	Improvement	p-value
Task A Time (sec)	245 ± 42	372 ± 58	34.1%	0.003
Task B Time (sec)	520 ± 89	780 ± 112	33.3%	0.004
SUS Score	78.4 ± 6.2	62.1 ± 8.4	+26.2%	<0.001
NASA-TLX (lower=better)	45.2 ± 7.1	58.6 ± 9.3	-22.9%	0.002
7-day Retention	84%	56%	+28%	—

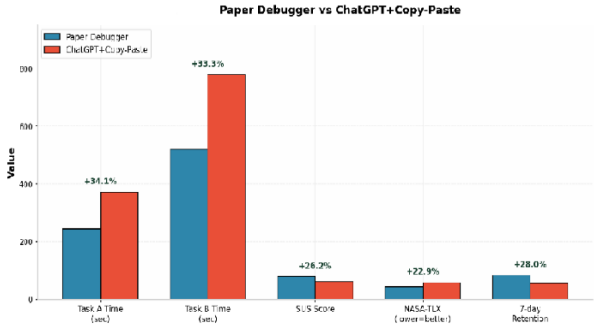

Figure 4: Comparison between ChatGPT-copy-paste and Paper Debugger.

The results indicate that this Paper Debugger significantly improves user efficiency across both experimental tasks. For Task A, participants completed the task in an average of 245 ± 42 seconds, compared to 372 ± 58 seconds using the ChatGPT copy-paste workflow, representing a 34.1% reduction in completion time. Similarly, for Task B, the average completion time decreased from 780 ± 112 seconds to 520 ± 89 seconds, corresponding to a 33.3% improvement. The associated p-values (0.003 and 0.004) confirm that these differences are statistically significant.

Usability was assessed using the System Usability Scale (SUS). The proposed framework achieved a score of 78.4 ± 6.2, substantially higher than the 62.1 ± 8.4 obtained by the baseline workflow. This represents a 26.2% increase in perceived usability, indicating that users found the integrated editor-native environment more intuitive and easier to use.

Cognitive workload was measured using the NASA Task Load Index (NASA-TLX), where lower values indicate better performance. Paper Debugger achieved a score of 45.2 ± 7.1, compared to 58.6 ± 9.3 for the conventional workflow, resulting in a 22.9% reduction in mental workload. This suggests that eliminating repetitive copy-paste interactions and providing integrated academic assistance reduces user effort and cognitive burden.

The final metric, 7-day retention rate, further demonstrates user preference for the proposed system. Approximately 84% of participants continued using Paper Debugger after one week, compared with 56% for the traditional workflow. This 28% increase in retention indicates stronger user engagement and greater long-term acceptance of the platform.

Overall, the results demonstrate that Paper Debugger significantly enhances productivity, usability, and user satisfaction while reducing cognitive workload. The statistical significance of the observed improvements provides strong evidence that integrating multi-agent academic assistance directly within the writing environment offers substantial advantages over conventional AI-assisted copy-paste workflows.

"Finally, Paper Debugger outperforms the ChatGPT+Copy-Paste workflow across all measured dimensions it's faster, easier to use, less mentally demanding, and leads to better learning retention."

8- THREATS TO VALIDITY

1- Internal Validity:

The within-subjects design may introduce learning effects. We mitigated this through counterbalancing and 48-hour washout periods between conditions.

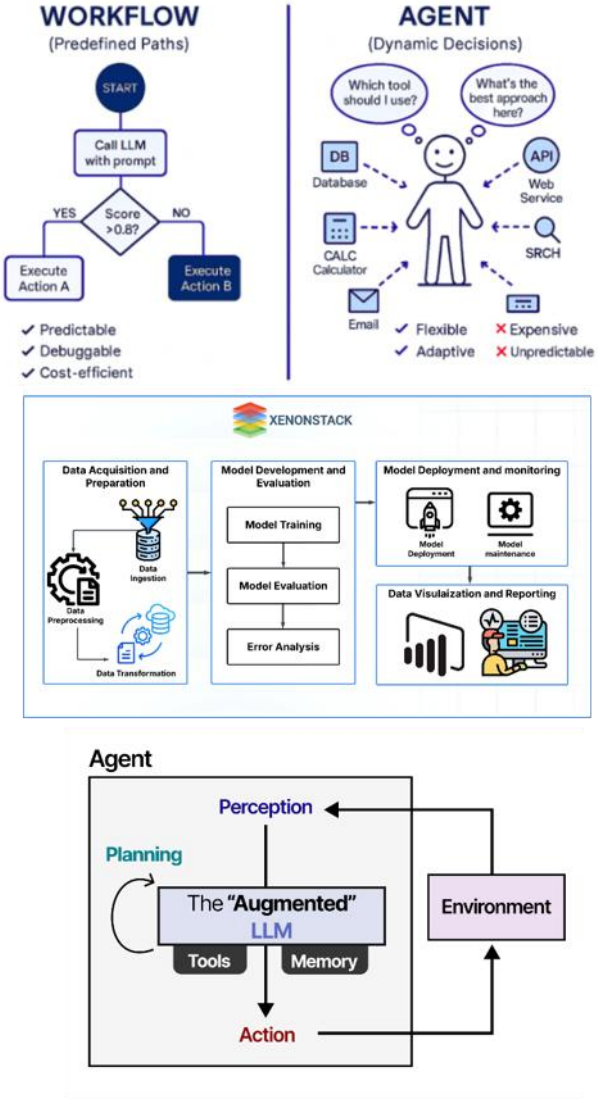

Figure 5: Multi-Agent Execution Pipeline.

2- External Validity

Our sample was limited to CS/Engineering researchers familiar with LaTeX. Results may not generalize to humanities scholars or Word users.

3- Construct Validity:

Task quality was assessed via automated tools; expert evaluation of a random 20% subsample confirmed consistency (Cohen's $\kappa = 0.82$).

Figure 5 depicts the multi-agent execution framework utilized in Paper Debugger. Incoming requests are directed to specialized agents skilled in reasoning, retrieval, and structured revision tasks.

9- DISCUSSION

The proposed framework advances prior research in AI-assisted writing by merging editor-integrated interactions with reasoning-driven capabilities. Unlike traditional writing tools that primarily emphasize language correction, Paper Debugger offers a cohesive environment that synthesizes retrieval, critique, verification, and revision processes. This innovative approach draws upon the latest developments in ReAct [6], Toolformer [7], Retrieval-Augmented Generation [8], and multi-agent systems powered by AutoGen [9].

9.1- SECURITY AND PRIVACY CONSIDERATIONS

Paper Debugger includes robust security measures to safeguard user data:

- Encrypted Communication: All communication between the Chrome extension and the Kubernetes backend is encrypted with TLS 1.3.
- On-Premise Deployment: Users can choose to deploy the entire system on-premise using Docker Compose or a local Kubernetes cluster, ensuring that document content remains within institutional infrastructure.
- No Persistent Storage: The system does not retain any document text permanently; all processing is conducted in-memory and discarded once the session ends.
- Cloud Data Retention: For cloud deployments, data is retained for a maximum of 24 hours before being automatically deleted.

These measures are designed to meet the stringent data protection standards commonly required by academic and research institutions.

9.2- LIMITATIONS

Paper Debugger, while offering notable strengths, has several limitations that warrant attention:

1. Network Dependency: The system requires a stable internet connection to interact with its backend and LLM services, rendering it impractical for offline use.
2. Hallucination Risk: As with all LLM-based systems, it may sometimes generate inaccurate or fabricated revisions, requiring users to carefully review suggested patches before implementation.
3. Language Support: Current capabilities are largely limited to English-language documents, with multilingual support planned for future development.
4. Context Window Constraints: Handling very lengthy documents—those exceeding 100 pages—can lead to increased latency due to context window limitations.
5. Browser Restrictions: Accessibility is currently restricted to Chrome users, as the system is available only as a Chrome extension.
6. Pilot Study Scope: The telemetry data presented in this paper is derived from a pilot deployment involving 25 users over 14 days. A broader, statistically rigorous user study is planned for future work to validate these initial findings.

10- CONCLUSION

This paper presents Paper Debugger, a multi-agent framework seamlessly integrated into the Overleaf editor to address the challenges posed by environmental gaps in AI-assisted academic writing. By embedding LLM-based reasoning directly within the editor, the system effectively mitigates workflow interruptions and reduces contextual discrepancies. Its architecture, supported by a Kubernetes-managed backend and a reliable patching mechanism, ensures that document modifications remain both scalable and transparent.

Pilot evaluation results suggest that the multi-agent approach reduces the need for manual copy-pasting and streamlines critique and revision workflows. Telemetry data from the initial deployment indicate high user retention rates, underscoring researchers' preference for a consistent, editor-native experience. Future improvements will focus on integrating automated fact-checking features to counter LLM hallucinations and expanding the framework to support collaborative, multi-author agent-based workflows.

11- REFERENCES

[1] [1] Lee et al., 2022 – CoAuthor: Human-AI Collaborative Writing
[2] Dang et al., 2022 – LLM Writing Support
[3] Zhang et al., 2023 – Essay Scoring with LLMs
[4] Amershi et al., 2019 – Guidelines for Human-AI Interaction
[5] Kulesza et al., 2015 – Explanatory Debugging
[6] Yao et al., 2023 – ReAct: Reasoning and Acting in LLMs
[7] Schick et al., 2023 – Toolformer
[8] Lewis et al., 2020 – Retrieval-Augmented Generation
[9] Wu et al., 2023 – AutoGen Multi-Agent Framework
[10] OpenAI, 2023 – GPT-4 Technical Report
[11] Madaan, A., et al. (2023). Self-Refine: Iterative Refinement with Self-Feedback. Advances in Neural Information Processing Systems (NeurIPS).
[12] Shinn, N., et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. Advances in Neural Information Processing Systems (NeurIPS).
[13] Khattab, O., et al. (2024). DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. arXiv preprint arXiv:2310.03714.
[14] Chase, H. (2024). LangChain: Building Applications with LLMs through Composability. GitHub repository.
[15] CrewAI Team. (2024). CrewAI: Multi-Agent Orchestration Framework for Autonomous AI Systems. GitHub repository.
[16] Writefull. (2024). AI-Powered Academic Writing Assistant. Writefull.com.
[17] Grammarly. (2024). GrammarlyGO: Generative AI Writing Assistance Platform. Grammarly.com.